Security and Privacy Controls in RFID Systems Applied to EPCglobal Networks

Wiem Tounsi

N° d'ordre : 2013telb0291

Thèse

Présentée à

Télécom Bretagne

en habilitation conjointe avec l'Université de Rennes I

pour obtenir le grade de
Docteur de Telecom Bretagne

Mention : *Informatique*

par

Wiem TOUNSI

Security and Privacy Controls in RFID Systems Applied to EPCglobal Networks

---

Composition du Jury :

Présidents : - M. Mohamed Mosbah, Professeur, ENSEIRB , Université Bordeaux 1

Rapporteurs : - Mme. Hakima Chaouchi, Professeur, Telecom SudParis
- M. Alban Gabillon, Professeur, University of French Polynesia

Examinateurs : - M. Frédéric Cuppens, Professeur, Telecom Bretagne (Directeur de thèse)
- Mme Nora Cuppens, Directrice de programmes, Telecom Bretagne (Co-directrice de thèse)
- M. Joaquin Garcia-Alfaro, Maître de conférences, Telecom SudParis (Encadrant)
- M. David Espes, Maître de Conférences, Université de Brest
- M. Matthieu Mallédant, Directeur d'entreprise, Télécom santé - Rennes

Acknowledgement

First, and foremost, I would like to thank my supervisors Nora Cuppens-Boulahia and Frédéric Cuppens for their inspiring support during the past years. Their rich experience and extensive expertise, combining with their passion for research, fill me with courage and motivate me to explore and conquer this world of scientific research. They have imparted the fundamental skills of our discipline and taught me how to further my skills independently. Their critical thinking and optimistic attitude towards work and life are rewarding for me forever. I am also indebted to Joaquin Garcia-Alfaro, for serving as examiner, who has helped me exploring ideas in RFID systems and express my results clearly by critically evaluating my work. His help and comments have been crucial to the establishment of many of the results presented in this thesis.

I am particularly grateful for the help I received from Yannick Chevalier at the university of Toulouse, who is one of the designers of AVISPA framework, especially for all the inspiring discussions on model checking techniques to formally verify the security properties of my protocol.

I am deeply obliged to my friends and colleagues in Telecom Bretagne for their support and help during my PhD program. I have learnt a great deal from them all. My thoughts go to Meriam, Mehdi, Sofien, Stere, Safa, Julien, Aymen and Ahmed who supported me from the first moments I arrived to the campus and provided me with a great comfort with their kindness, generosity and humor. My deepest gratitude goes to Mariem for her special friendship. She is an endless source of advice on topics as diverse as academic life, personal life and culinary master-pieces. All my thankfulness is for the wonderful members of our team (SFIIS, now), for always contributing to create a pleasant and stimulating group atmosphere; my thoughts go to Nada, Benjamin, Samiha, Said, Sabeer, Fabien, Tarek, Hanieh, Anis and Pierre.

Finally, I am deeply indebted to my family: my extraordinary parents for supporting and advising me in every thing I have chosen to do. Their unflagging care, endless love and faith in me have always been a beacon of confidence and a source of perseverance ; To my sister and brother who fill me with true love and with whom I always have much fun; To my grand-parents who continuously support me and resource me with their affection and sweet feelings. Last but not least, I deserve special acknowledgement to Rami, my significant other whose trust and sacrifice motivated me to focus on the accomplishment of this thesis. Our enjoyable travels have gratefully inspired me, with all the unforgettable moments of fun and humor we have shared.

I am deeply glad to keep the best memories with all of you.

Abstract

Radio Frequency IDentification (RFID) provides a way to automate identification and to store information in individual RFID tags. These tags can be attached or embedded in an item to be identified and are read when they enter an RFID reader's antenna field. The Electronic Product Code (EPC) Class 1 Generation 2 (Gen2 for short) is a proper example of passive RFID technology. It represents the key component of an RFID architecture named EPCglobal network. However, if the tag carries more than just an identifier, the privacy of the tag holder may be violated. In this thesis, we deal with privacy issues in two levels of the EPCglobal network to only let authorized entities access private data. Our goal is to ensure that the data exchange from RFID tags to middleware and enterprise applications guarantees the privacy requirements, in environments where privacy control is paramount, e.g., home healthcare monitoring systems.

The first part of this dissertation is dedicated to securing data exchange between RFID readers and passive tags. We provide a key establishment and derivation protocol for Gen2 systems, called KEDGEN2, to handle the flawed security model of the Gen2 tag memory access. KEDGEN2 achieves secure data exchange, based on a key generation model adapted to Gen2 tags. To prove the security of our model, we specify the protocol using the High Level Protocol Specification Language and verify the expected security properties, using the Constraint-Logic based Attack Searcher model checking tool. The current version of the protocol guarantees mutual authentication of participants and forward secrecy in the presence of active adversaries. It also guarantees backward secrecy with active adversaries bounded by limited communication range, which is consistent with typical RFID environments. As for derived keys, we propose adapting the Solitaire cipher, as a Pseudo-random Number Generator.

To complement our approach, an additional filter is added and described in the second part of this dissertation. We focus on the collection of tag information through the RFID middleware component. The middleware is a central point that sits between RFID readers and database applications. It is in charge of collecting, filtering and aggregating the requested events from heterogeneous RFID environments. Thus, the system at this point is likely to suffer from parameter manipulation and eavesdropping, raising privacy concerns. We propose a privacy-enhanced approach as a part of the RFID middleware of the EPCglobal network, which does not interfere with the standard interface. Our approach is policy driven using some enhanced contextual concepts of the extended Role Based Access Control model. We use specifically, the PrivOrBAC privacy-aware model to store and manage privacy preferences, taking the declared purpose, the accuracy and the explicit consent, as privacy requirements. To show the feasibility of our approach, we provide a proof-of-concept prototype that we apply to the Fosstrak plateform, an open-source implementation of the EPCglobal specifications.

Résumé

La technologie Radio Frequency IDentification (RFID) est un moyen d'automatiser l'identification et le stockage des informations dans des étiquettes RFID. Ces étiquettes peuvent être fixées ou intégrées à un objet à identifier et sont lues par un lecteur RFID. Electronic Product Code (EPC) Classe 1 Génération 2 (Gen2) est un exemple typique des étiquettes RFID passives. Il représente l'élément clé d'une architecture RFID, nommée EPCglobal network. Ces étiquettes peuvent contenir plus d'informations qu'un simple identifiant, comme les données personnelles du porteur de l'étiquette. Par conséquent, des protections sont nécessaires pour protéger la vie privée du porteur. Dans cette thèse, je propose un contrôle des données à caractère privé à deux niveaux de l'architecture EPCglobal, pour que seules les entités autorisées puissent récupérer ou modifier ces données privées. L'objectif est d'assurer que l'échange de données depuis les étiquettes RFID vers les *middleware* et applications d'entreprise répond aux exigences de confidentialité, dans un environnement où le contrôle de la vie privée est primordial, par exemple, les systèmes de suivi de la santé à domicile.

La première partie de la thèse est dédiée à la protection des échanges de données entre les lecteurs et les étiquettes RFID passives. Je présente un protocole d'établissement et de dérivation de clés dans les systèmes Gen2, appelé KEDGEN2. Il traite les insuffisances du modèle de sécurité du standard quant à l'accès à la mémoire de l'étiquette. Le protocole a été spécifié en utilisant le langage HLPSL (High Level Protocol Specification Language) et des propriétés de sécurité (authentification mutuelle, *forward secrecy* et *backward secrecy* pour la persistance de la confidentialité) ont été prouvées grâce à des techniques de *model checking* via l'outil CL-AtSe (Constraint-Logic based Attack Searcher). Le mécanisme de dérivation de clés pseudo-alétaoires repose sur une adaptation du système de chiffrement Solitaire.

Pour compléter notre approche, la deuxième partie de cette thèse ajoute un filtre au niveau des données collectées par l'intergiciel RFID (appelé *middleware*). Ce composant central se situe entre les lecteurs d'étiquettes et les bases de données applicatives. Il est en charge de collecter, filtrer et agréger des événements issus d'environnements RFID hétérogènes. L'approche que je propose est centrée sur l'utilisateur, qui peut définir la politique de vie privée s'appliquant à ses données agrégées, sans interférer avec l'interface standard du *middleware*. Elle permet l'expression des préférences de vie privée en utilisant le modèle PrivOrBAC avec la prise en compte des dimensions de déclaration de l'objectif, de précision des résultats générés, et de consentement explicite de l'utilisateur. Un prototype a été développé pour illustrer la faisabilité de l'approche et a été appliqué à l'infrastructure open-source Fosstrak.

Contents

CHAPTER 1

Introduction

"Concern for man himself and his safety must always form the chief interest of all technical endeavors. Never forget this in the midst of your diagrams and equations." – Albert Einstein

Radio Frequency IDentification (RFID) has emerged as a low-cost technology due to the use of battery-less devices. It is acclaimed as the successor of today's ubiquitous barcodes. Contrary to optical barcodes, RFID devices allow the identification of individual objects much faster, without a line of sight and human intervention. These devices are known as passive RFID tags. They can be attached or embedded in an item to be identified and are read when they enter a RFID reader's antenna field. The relatively low-cost of the passive RFID technology makes it attractive to logistics departments that search for possibilities to integrate this real-time technology in their business processes. In addition, it was shown that this technology significantly improves the visibility and accuracy of the logistic operations [130]. Despite its wide use and prospective applications, RFID technology poses several security and privacy issues, which could adversely affect its global adoption.

1.1 Context and Motivation

RFID technology was originally envisioned to automate data collection in a supply chain. Over the last years, substantial progress has been made to integrate this technology into context-aware applications. A promising scenario is the use of passive RFID tags to implement home healthcare systems. Indeed, using RFID technology in such critical domain has grown in importance with the need of assisting people (e.g., elderly individuals and patients) in their everyday life [82, 94]. Hereafter, we present the facts. According to a study made at U.S. Office of Censorship in 2000 [137]: *the net balance of the world's elderly population has increased by more than 750,000 per month; Two decades from now, the increase will likely be 2 millions per month.* Also, it was noticed in [28], that elder patients, suffering from chronic disorders are the most subscribed to home healthcare services. In addition, a crucial finding in [60] was that patients stressed the need to feeling in control, apart from their desire to be surrounded by a pleasant atmosphere. The aforementioned reasons show the patient need to be remotely assisted by efficient monitoring systems. On the other hand, nurses, who should regularly visit their patients can rely on these new systems to ease their workload and reduce administrative tasks [109]. For example, the use of RFID technology helps ensuring that the right drug is administrated to the patient at time and in the right dosage. As a result, more

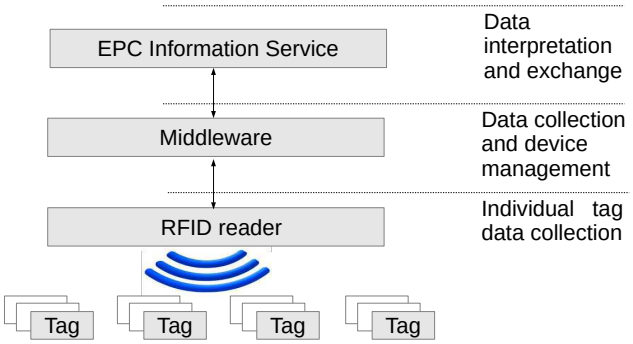


Figure 1.1 – Main levels in EPCglobal network

and more healthcare systems have made use of new technologies such as low-cost RFID, to help improving their quality and providing substantial cost saving. But still, these healthcare solutions need to be carefully designed to guarantee the minimum of privacy.

The deployment of the passive RFID technology is becoming more important with the standardization process through the Electronic Product Code (EPC) Class 1 Generation 2 tag standard [61] (known as Gen2). Designed in the Massachusetts Institute of Technology, and developed by the EPCglobal consortium [65], the EPC technology represents the key component of an architecture named EPCglobal network [61]. EPCglobal standardization covers the whole RFID architecture [11], from tag data structure to network communication specifications. The main goal of this architecture is the object automatic identification and memory reading to share this information through the business process.

The EPCglobal network is a set of global technical standards consisting, from a highly abstract view, of three basic levels, cf. Figure 1.1: the radio level linking RFID readers to tags, the middleware real time data and devices management level and the EPC Information Service (EPCIS) level. EPCIS deals with filtered and aggregated data to translate them into the related business events and makes them available to internal and external accesses. Thus, persistence is provided in this level by an EPCIS repository maintaining historical events. Despite their wide use and prospective applications, RFID technologies pose several security and privacy issues, which could adversely affect their global adoption. It is worth noting that RFID technology, by its design, is not privacy friendly, since its original goal is to enable fast and automated object identification.

As in many other emerging technologies, if countermeasures against attacks (e.g., data leakage, impersonation) are not handled properly at the lowest level of the architecture, security risks and privacy violations happen. In this dissertation, we focus on the use of passive RFID devices, specifically the Gen2 tags. We deal with data leakage as a privacy issue for each of the insecure radio communication interface linking reader to tag and the middleware collection level. Data leakage involves the disclosure of the tag identification (i.e., the EPC, in Gen2 technology) and the content of the tag additional memory. These data raise a real concern, if they are linked to personal information, e.g., a bearer's illness or age.

In this context, many challenges need to be addressed. In the following section, we detail the ones that we focus on in this manuscript.

1.2 Challenges

The challenges that need to be addressed in this thesis could be organized in two parts. The first part concerns the insecure communication protocol linking readers to tags in the RFID front-end infrastructure. The second part focuses on the privacy issues raised in the back-end of the same infrastructure, particularly, the RFID middleware for filtering and collecting events.

First challenges Providing security for the reader and tag communication, with respect to passive RFID devices is a challenging task. On the one hand, the standard provides very basic protection for privileged operations that require reader authentication (i.e., access the tag memory or deactivate the tag). On the other hand, the capabilities of RFID Gen2 tags are very limited to embed traditional cryptography on-board. At the same time, it is of utmost importance to keep the cost of RFID Gen2 technology low to promote its wide use.

Regarding the first point, in the last RFID standard ratified by EPCglobal, tags support on-chip pseudo-random number generator (PRNG) and Cyclic Redundancy Check (CRC) computations. Privacy protection mechanisms are to make the tag permanently unusable once it receives the *kill* command with valid 32-bit kill password. The privileged access such as read/write to tag memory is allowed only after the tag is in secure mode. This mode is reached upon the tag receives an access command with a valid 32-bit access password (divided in two messages). At this point, a real threat raises due to the ineffective model used to protect the access password. In fact, this latter password is sent from the reader applying an Exclusive-OR operation with a pseudo-random key. This key is previously generated in the same session and sent in plaintext from the tag.

It is straightforward that an adversary capable of eavesdropping the pseudo-random key sent by the tag using special hardware devices (e.g., readers with high sensitive receivers and multiple antennas), or predicting the output of the random bit generator of the tag (e.g., based on a flawed EPC Gen2 pseudorandom generator [120]), can obtain this key. Thus, it can simply recover the access password by applying an Exclusive-OR operation. Obtaining the access password permits to the malicious adversary to access and modify the memory of the tag. When the memory of the tag includes additional information about objects which can be linked to people holding them (e.g., information related to a particular medicine which may link to a particular disease), information disclosure can be an important threat. Such a threat can be handled by modifying the security model of the Gen2 specification and providing a more elaborated way of managing cryptographic keys in Gen2 systems. Hence, the problem turns to be a key establishment concern to ensure that a new encryption key is

used in every new session. This key serves as a master key for subsequent derivations (i.e., key generation) of keys within the same session.

Regarding the second point, despite the efforts made to adapt hash methods or public key cryptography for low-cost devices, such approaches are beyond current capabilities of passive RFID tags [26]. As a result, cryptographic protocols for passive RFID tags can at best be built using symmetric primitives based on key generation functions. Thus, the next security challenge is related to the key generation function. RFID technologies based on lightweight PRNGs should not output numbers with poor statistical properties by using predictable secret keys (e.g., first propositions to protect RFID tag exchanges [70, 160]). Our aim is then to use a lightweight PRNG and to maximize the chance of generating random looking keys. This security problem turns to be a privacy protection as well, since the RFID tag can combine the sensitive data with a different key in every transaction.

It is important to note that even when using a strong cryptographic algorithm as part of the communication protocol, this does not guarantee by itself the security of the protocol. Indeed, good cryptography can be used in a bad way due to a bad designed protocol. Many protocols were proposed in the literature, however, most of them turned out to be flawed few time after their publication. The flaws were typically related to the protocol. The most well-known case is WEP (Wired Equivalent Privacy), the first security protocol for WiFi networks. The RC4 stream cipher used in WEP is remarkably simple to implement and considered to be *strong* if used in a proper way. Nevertheless, the designers of WEP did not do so. As many analyses conclude, the weaknesses of WEP do not derive from faults in RC4, rather from the way it is applied [59]. Thus, a major need of proved and validated security protocols appears. These proofs may be done following a computational or a symbolic approach. These proofs can be aided by automatic tools, which are developed for this purpose. Unfortunately, this was not systematic in most works dealing with RFID security where the security goals were demonstrated *intuitively*. This formal verification of the protocol represents another challenge, where mutual authentication of the tag and the reader and strong notions of secrecy have to be analyzed.

Second challenges After being acquired by the RFID reader, the data collected from tags are transferred to a back-end infrastructure for further processing (e.g., filtering, aggregation). Thus, they can potentially be learned by multiple parties, such as curious applications. This presumably allows for a fine-grained and even excessive surveillance mechanisms that would pervade the patients lives [44]. For instance, according to the Health Information Portability and Accountability Act (HIPAA) guidelines [10], dealing with privacy concerns means that a collector must not be able to identify the product type by simply reading the RFID tag's ID code. Thus, the identification could be realized only by considering some parts of the ID code such as a product manufacturer or a family of tags depending on the requester purpose. The same issue is raised when some fields in the tag memory are read. If this control could be held before the generated events are sent to upper layer applications, where

information is interpreted and stored, it would be interesting that the RFID middleware, which is responsible for configuring readers, considers these consumer concerns as well as the legal guidelines. The middleware sits between the reader and repositories applications, cf. Figure 1.1. It is the central point for both data collection and request configuration. As a consequence, the system at this point, is likely to suffer from parameter manipulation and eavesdropping leading to privacy concerns.

Most middleware implementations available today propose a role based access control as a security approach. However, they fail to address consumer privacy concerns by supporting appropriately the fair information practices [67]. A possible reason for this issue is that most enterprises run the RFID middleware in an internal network [39]. Besides, as explained in [34], the effort to achieve security and privacy exists in current works on RFID middleware, however, it is not as an architectural incorporation, rather as domain specific security rules, implemented up to compliance to specific middleware. The final challenge, in this thesis, is to show the possibility of modifying the existing EPCglobal middleware standard in order to satisfy the basic principles of privacy. The aim is to propose an approach that applies to all applications where critical information is to be preserved securely. For example, applications in the context of home healthcare.

1.3 Contributions

The main contributions of this thesis are summarized in five points:

New key establishment and derivation protocol for passive RFID devices: We propose a new key establishment and derivation protocol for Gen2 systems designed to share and refresh keys between readers and tags. Our new protocol, namely KEDGEN2 focuses on the privileged access to the tag memory, where keys and other sensitive information could be stored. We propose to use an alternative model to the EPC Gen2, assuming the use of a pseudo-random number generator (PRNG) whose algorithm and internal states are known at both reader and tag sides (cf. [171, 175]).

Adaptation of the Solitaire keystream: Since it has been shown that the passive RFID tags have similar capabilities to humans [91] and due to the low complexity required by the card shuffling in Solitaire keystream [161], we propose to adapt the hand cipher Solitaire as a PRNG. The generated keys are used in the protocol as a one time encryption keys. To find the generation threshold in order to update the internal state, we search for the cycle size of the Solitaire states, named the orbit size in group theory. Differently from classical generators, we found that the orbit size in Solitaire is variable depending on the cards positions (the internal state). To adapt it to KEDGEN2, we modify it by updating the initial state (the seed) with respect to the maximum orbit size. Finally, we demonstrate that the modified version of Solitaire can be easily integrated in our KEDGEN2 protocol (cf. [176]).

Formal verification of the KEDGEN2 protocol: Even when using a strong cryptographic algorithm as part of a communication protocol, this does not provide guarantees that the protocol will be secure. To verify the security of KEDGEN2 protocol with respect to strong notions of secrecy and authentication, we follow a formal approach. More specifically, we use a model checking tool, which is a common formal verification technique using a symbolic approach. The Constraint-Logic based Attack Searcher (CL-AtSe) which is known for its maturity in constraint solving techniques [99] is adopted. The current version of the protocol successfully handles the properties of mutual authentication and forward secrecy under a strong adversary model. The backward secrecy is also guaranteed under a modified but realistic adversary, which captures the capabilities of a real world adversary in typical RFID environments. Forward secrecy (respectively, backward secrecy) means that after the exposure of a given master key, the adversary cannot compute previous (respectively, future) master keys used in the system once the master key is refreshed (cf. [174]).

A policy-driven model to enforce privacy in EPCglobal middleware: It is uncontested that the act of reading out one or more RFID tags constitutes a data collection. Thus existing privacy laws and regulations apply to the communication involving the RFID middleware. We propose a privacy controller module that enhances the filtering and collection middleware of the EPCglobal network. Our privacy model is policy-driven using some enhanced contextual concepts of the extended Role Based Access Control model. It ensures the following features: (i) enforcing a privacy policy without interfering with the standard middleware interface, (ii) using an existing privacy-aware model to store and manage privacy policy preferences, and (iii) taking into account the principles of purpose, consent, and collection limitation (accuracy) as privacy requirements (cf. [173]).

A proof-of-concept prototype implemented in the Fosstrak framework: To show the feasibility of our privacy-enhanced model, we provide a proof-of-concept prototype integrated into the middleware of the Fosstrak framework. Fosstrak [69] is an open-source implementation of the EPCglobal specifications. It is a recommended implementation of the EPCglobal providing an EPCglobal-certified EPCIS Repository [65] (cf. [173, 172]).

1.4 Organization

The remainder of the manuscript is composed of two parts. The first part deals with the insecure communication protocol linking readers to tags in the RFID front-end infrastructure, whereas, the second part handles the privacy issues raised in the RFID back-end middleware for filtering and collecting events.

Chapter 2 provides a comprehensive background on EPC Gen2 technology, particularly the proposed security protocol and the on-board pseudo-random number generator (PRNG). It also gives an overview of cryptographic primitives with respect to the limited environment

of RFID devices as well as a discussion of main previous work on PRNG constructions. Moreover, this chapter introduces the basic techniques of key establishment and derivation protocols and presents relevant work on RFID protocols. Finally, it reviews the formal verification approaches of security protocols, including the model checking tool used in this dissertation.

Chapter 3 presents a new key establishment and derivation protocol for RFID Gen2, that we name KEDGEN2. The adversary model and the targeted security protocols are provided at first. Then, the description of the protocol components, sessions and stages is given. This chapter also provides the Solitaire keystream generator as a part of the protocol to generate the one time keys. The cycle detection results of Solitaire outputs and the estimated cycle size are given. Finally, this chapter demonstrates that the modified version of Solitaire can be easily integrated in the KEDGEN2 protocol for the generation of the derived keys.

Chapter 4 deals with a formal approach to verify the KEDGEN2 protocol with respect to strong notions of secrecy and authentication. Using the CL-AtSe model checking tool, we prove formally that after being amended, the current version of the protocol successfully handles the mutual authentication and the forward secrecy properties under a strong adversary. The backward secrecy is also guaranteed assuming a weaker but realistic adversary model, consistent with typical RFID environments. The chapter is ended by a comparative discussion between relevant related work and our proposal.

Chapter 5 provides more comprehensive background on the backend architecture of the EPCglobal network with a focus on the middleware role and its interface. This interface comprises a reading API which is responsible for configuring and receiving the collected RFID data. It also includes an access control API whose limitations regarding privacy control are shown. In addition, relevant works on securing the data processing in the RFID middleware are presented. This chapter also surveys the privacy dimensions appearing in known guidelines and regulations and presents the relevant privacy models based on access control. A particular focus on the PrivOrBAC [8] model issued from Organization-Based Access Control model (OrBAC) is done. This model handles most of the privacy dimensions appearing in the guidelines and regulations that will be presented.

Chapter 6 shows how the existing interface of the EPCglobal middleware standard can be modified to satisfy the privacy principles of declared purpose, consent, and collection limitation (accuracy). A motivating example is detailed and a related privacy policy specification is provided. The chapter ends by showing that the privacy integration can be handled in the reading API of the middleware interface, without interfering with the standard interface.

Chapter 7 shows the feasibility of our policy driven approach to enforce privacy in the middleware level. A proof-of-concept is provided showing the successful integration of our privacy module into the open-source software for track and trace Fosstrak. Finally, some performance analysis of our proposal are given.

Chapter 8 concludes the manuscript and provides our perspectives for future work.

Part I

A Key Establishment and Derivation Protocol for Passive RFID Tags

CHAPTER 2

State of the Art

"The seeker after truth is not one who studies the writings of the ancients and, following his natural disposition, puts his trust in them, but rather the one who ... questions what he gathers from them, the one who submits to argument and demonstration." – Ibn al-Haytham

Introduction

Passive Radio Frequency IDentification (RFID) devices are gaining a prominent place in several domains [86, 153], even those dealing with sensitive information, such as hospitals and inpatient care systems [186]. The EPC Gen2 standard, short-hand for the Electronic Product Code (EPC) Class-1 Generation-2 [61], is a proper example of passive RFID technology. Although the RFID technology offers several benefits, its deployment still presents a variety of security and privacy implications. In this chapter, we are interested to the interface linking EPC Gen2 tags to readers. The privacy implications at this level are related to both the security protocol to access the tag memory and the on-board key generator used for each transaction. To this end, we first introduce the EPC Gen2 technology characteristics, with a focus on the EPC Gen2 tags protocol and key generator models. A comprehensive background on the symmetric cryptography, which is mostly used in passive RFID tags communications is thus given. Next, we provide a brief overview of relevant work on Pseudo-Random Number Generators for constrained RFID tag. We also survey some relevant key establishment protocols for passive RFID systems. Finally, we close this chapter with an overview on security protocol verification approaches.

2.1 EPC Gen2 Standard

The EPC Gen2 standard, short-hand for the Electronic Product Code (EPC) Class-1 Generation-2 [61], is a passive RFID technology. Designed in the Massachusetts Institute of Technology, and developed by the EPCglobal consortium [65], the EPC technology is intended to be the successor of the nowadays ubiquitous barcodes. EPC Gen2 scenarios are designed with very basic capabilities. As other communication models using low-cost RFID systems, Gen2 tags derive their transmission and computational power from the signal of an

interrogating RFID reader. They work worldwide on the ultra high frequency (UHF) band between 860 and 960 MHz, depending on the RF regulations for each continent. In a low-cost RFID system, like EPC Gen2, the tags are resource limited, allowing to reduce their cost under the 10 cents of US dollar [162]. This reduction on the tag cost is proportional to the size of the silicon Integrated Circuit (IC). The typical measure of space in silicon ICs is the gate equivalent that is equivalent to a boolean of two-input NAND gate. The estimations on available gates for EPC Gen2 implementations are around 10 K gates [43].

Once a scan is done by the RFID reader, the tag responds with a unique ID, called EPC, that is transferred to a back-end infrastructure for further processing (cf. Part II of our dissertation). The EPC number serves as an identifier for the physical objects, locations and assets carrying the tag which, moreover, can be identified and tracked based on completely distributed architectures [110]. In this first part of the dissertation, we focus on the communication between the tag and the reader. Before delving into the details of this two-party communication, a presentation of the components of an EPC tag is provided hereafter.

2.1.1 Structure of the Electronic Product Code

Information about EPC numbers is not stored in the code itself, but serves as a reference to Internet-based information. Moreover, other information may be stored in the additional memory of the tag, depending on the tag type. The EPC Gen2 standard defines the identification sequence with 96 bits [61], but other identification sizes can be used depending on the tag manufacturer. For instance, the format provided in Figure 2.1 is only one possibility.

The code of the EPC is divided into four, fixed length partitions, as shown in Figure 2.1. First, the header defines the number, length and type of subsequent data partitions. Second, the EPC manager typically defines the manufacturer/company or creator of the item. It maintains the domains of both object type codes and serial numbers. The third position occupies the object type code, and the last position defines the unique object identification number.

Since, the EPC was not made to replace but to integrate existing identifier schemes, different EPC headers can be used for different schemes. This includes the following GS1 identifiers: Serialized Global Trade Item Number (GTIN), Serial Shipping Container Code (SSCC), Global Reusable Asset Identifiers (GRAI), Global Individual Asset Identifiers (GIAI), and Global Location Number (GLN). The EPC Tag Data Standard [76] defines the structure of the EPC for each identifier scheme and provides the encoding/decoding rules, while EPC Tag Data Translation Standard [75] provides this in a machine-readable format, to support translation at any stage in the EPC network architecture. For more details about the data representation in each stage, we refer the reader to Appendix B.

In the remainder of this section, we introduce in more detail the properties of the communication between tag and reader and its security model.

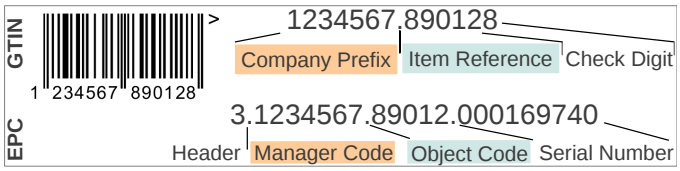


Figure 2.1 – EPC structure and GTIN integration

2.1.2 Tag and Reader Communication

EPC Gen2 tags do not have a power source. Instead of this, tags are passively powered by the reader and can only respond after receiving a message from the reader. The communication is half-duplex thus, simultaneous transmission and reception are not allowed. The communication between tag and reader in the EPC Gen2 system is organized in three stages: *Selection*, *Inventory* and *Access* stages. In the *Selection* and *Inventory* stages, the reader initiates the communication sending identification queries. If the reader manages to access or modify the tag memory content, the *Access* stage is started. Starting from this point, an important security drawback is encountered, as the required password for the access is not sufficiently secure. The communication in the EPC Gen2 protocol is organized as follows:

- *Select*: The reader selects a part of the tag population in the communication neighborhood for inventory and access using one or more *Select* commands.
- *Inventory*: It is the process by which a reader identifies tags. A detailed description of the inventory operation is as follows:
  1. A reader sends a request message *Query* to a tag. The query initiates an inventory round and decides about tags that will participate in the round.
  2. The available tags in the communication range pick a 16-bit random value using the implemented PRNG on-board, and shall load this value into a slot counter, decreasing one unit for each *Query* command reception. When the slot counter reaches zero, the tag backscatters a random value (named RN16) to the reader.
  3. If the reader detects a single tag reply, it requests the identification from the tag by acknowledging the tag with an ACK containing the same RN16.
  4. The tag compares the RN16 in the ACK with the RN16 it sent before. If they are equal, the tag backscatters its EPC (Electronic Product Code), PC (Protocol-Control), and CRC-16.
- *Access*: After acknowledging a tag, a reader may choose to access it. In this stage, a reader modifies or reads individual tags' memory areas. A reader can access a tag as follows:
  1. The reader sends a *ReqRN*, containing the previous RN16, to the acknowledged tag.

2. The tag compares the random number RN16 in the *ReqRN* with the RN16 in the tag. If it is right, the tag generates and stores a new RN16, denoted as handle. Then, backscatters it to the reader.
3. The Read, Write, and Kill commands are sent with 16-bit words (i.e., either data or half-passwords) from reader to tag. These commands use a one-time-pad key to hide the word being transmitted.

2.1.3 Flawed EPC Gen2 Security Model

As in many other emerging technologies, if countermeasures against attacks are not handled properly at the lowest level of the architecture, where the exchange of information between Gen2 tags and readers is carried via the insecure radio interface, security risks and privacy violations happen. However, the development of Gen2 security countermeasures faces several challenging constraints such as cost, power consumption, and performance requirements. From the approximately ten thousand available gates in EPC Gen2 integrated circuit, only two to five thousand can be devoted to security tasks [43]. Thus, on-board security tools must be necessarily simple. In the sequel, we first detail the Gen2 security protocol, then we deal with the on-board pseudo-random generator.

Security Protocol

The security protocol of the EPC Gen2 communication only considers a basic protection for some special operations that require reader authentication, such as memory access and deactivation operations. The EPC Gen2 standard includes in its specification a 32-bit password to protect the tag memory access and a 32-bit password for the *kill* command execution. This command allows to permanently deactivate the tag, or to unblock specific tag memory areas previously blocked (*recomission*), depending on the command codification [61]. The kill and access passwords are stored in the tag reserved memory area. This password must be sent via the insecure reader-to-tag channel. Since this channel is more likely to suffer from eavesdropping attacks than the low-power tag-to-reader channel, the Gen2 specification proposes to protect the exchange by relaying on the tag-to-reader channel. Using this channel, the tag transmits in plaintext the nonces to be used as a keystream to protect the reader-to-tag communication. The model is detailed in the following steps:

1. READER $\longrightarrow$ TAG : Key-request
2. TAG $\longrightarrow$ READER : Keystream
3. READER $\longrightarrow$ TAG : Password $\oplus$ Keystream

In Step 1, the reader informs that it is waiting for a key necessary to protect the sensitive data in the upcoming exchange (cf. Step 3). This exchange will eventually contain the required password to grant the execution of an *access* operation. The key is generated by the tag as a random bit string, and transmitted in Step 2 in plaintext to the reader. This is

done via the tag-to-reader channel which, in principle, is expected to have an eavesdropping range much lower that the reader-to-tag channel. The exchange supposes that an adversary eavesdropping the tag-to-reader channel cannot capture the sensitive data (i.e., either the password or the contents of the password-protected operation).

It is straightforward that an adversary capable of eavesdropping the tag-to-reader channel using special hardware devices (e.g., readers with high sensitive receivers and multiple antennas), or predicting the output of the random bit generator of the tag (e.g., based on a flawed EPC Gen2 pseudo-random generator [120]), can obtain the key and simply recover the access password by applying an Exclusive-OR operation. Obtaining the access password permits to the malicious adversary to access and modify the tag memory. When the memory of the tag includes additional information about objects which can be linked to people holding them (e.g., information related to a particular medicine which may link to a particular disease), information disclosure can be an important threat. Such a threat can be handled by modifying the security model of the Gen2 specification and providing a more elaborated way of managing cryptographic keys in Gen2 systems. This is the subject of our first contribution that will be detailed in the next chapter. We present in the next paragraph the pseudo-random number generator characteristics of the EPC Gen2 technology.

On-board Pseudo-Random Number Generator (PRNG)

Traditional keystream generators share a secret k for the *one-time pad* communication between sender and receiver. The use of deterministic PRNGs to generate exactly the same sequence in both sender and receiver sides is commonly used. However, in the specific communication model of EPC Gen2 systems the sender and receiver cannot share any secret k. Instead, the low-power tag-to-reader channel is used to transmit in plaintext the nonces to be used as a keystream for the reader-to-tag communication. The on-board PRNG generates 16-bit pseudo-random sequences, and shall have the ability to provide RN16s to the tag anti-collision system, to acknowledge other Gen2 specific operations (e.g., memory writing, decommission of tags, and self-destruction). PRNG is, therefore, a crucial component that guarantees Gen2 security. More details on PRNG will be given in Section 2.3. As specified in the standard, the PRNG is supposed to meet three randomness criteria. We name (1) the probability for the single 16-bit sequence j drawn from the generator to be bounded by $\frac{0.8}{2^{16}} < Prob(j) < \frac{1.25}{2^{16}}$, (2) the probability that any two tags simultaneously generate the same 16-bit sequence shall be less than 0.1% for a tag population of up to ten thousand tags, and (3) the chance of guessing the next 16-bit sequence generated by a tag shall be less than 0.025% even if all previous outputs are known to an adversary.

Existing commercial EPC Gen2 tags implement an on-board PRNG, as required by the EPC specifications [61]. However, companies do not provide their designs [145]. They simply refer to testbeds that show some compatibility requirements. This method, mostly called security through obscurity, is always ineffective in security engineering, as it has been shown

with the disclosure of the vulnerable PRNG of the Mifare Classic chip [70] (i.e., it was shown that the PRNG internal state is always initialized with the same values. Thus, a synchronization attack with precomputed values suffices to predict the the secret of the stream cipher) . This drawback constitutes the motivation of our second contribution that proposes a new PRNG suitable to Gen2 communication. This contribution will be presented in the next Chapter.

In the sequel, we give an overview of symmetric cryptography especially used for limited environments such as constrained RFID devices. We also provide a brief relevant work in Pseudo-Random Number Generators with respect to the constrained passive RFID tags.

2.2 Symmetric Cryptography

2.2.1 Security Goals

The basic cryptography and the RFID security goals are detailed hereafter.

Basic Cryptography Goals

The application of cryptographic primitives in information systems is based on the following goals [121]:

- *Confidentiality* is used to keep information accessible only to the authorized entities. There are many approaches to provide confidentiality. They range from physical protection to mathematical algorithms which render data unintelligible.
- *Data integrity* handles the unauthorized alteration of data. To assure data integrity, the system should be able to detect data manipulation by unauthorized parties. Data manipulation includes insertion, deletion, and substitution.
- *Authentication* is closely related to identification. This function applies to both entities and information itself. Two parties entering into a communication should identify each other. Information delivered over a channel should be authenticated as to origin, date of origin, data content, time sent, etc. Data origin authentication implicitly provides data integrity.
- *Non-repudiation* prevents the sender or the receiver of a message from denying previous actions.

RFID Protocol Security Goals

Security protocols, aiming at securing communications are designed to satisfy the aforementioned goals, e.g., *Confidentiality* (to ensure *Secrecy* of secrets) and *Authentication* are fundamental security properties, which should be conisdered by RFID systems. In addition to these security goals, an RFID protocol generally requires other properties, known as *Anonymity*, and *Untraceability*.

- *Anonymity* allows an entity to make transactions that cannot be known by others. If the tag ID can be kept anonymous, the problem of leaking information pertaining to the user belongings would be solved. Anonymity is the topic of most researches in the RFID literature (cf. Section 2.4.2). Intuitively, it says that an attacker cannot discern the tag from any information revealed by the owner during the identification stage. The problem of identifying the type of tag can be treated by using pseudonyms. This solution may be sufficient in contexts like healthcare monitoring systems to enforce privacy, since it does not allow a rogue reader to recognize the type of medicine or object used by the patient. In the literature (e.g., [88]), a stronger solution is applied to deal with this problem. It relies on multiple pseudonyms assigned to one tag. The goal is to make tracking harder for the adversary. Each time a tag is queried, it releases one of the pseudonyms from the list it possesses, cycling to the beginning when the list is exhausted. That is, only a valid reader (or a determined adversary) can tell when two different pseudonyms belong to the same tag.

- *Untraceability* assures that an attacker cannot trace the movement of a tag. RFID readers in strategic locations can record observations of unique tag identifiers. These identifiers could be associated with personal identities. As passive tags respond to readers without alerting their owners, the threat comes out when a tag identifier is linked to personal information. For example, by using an e-passport, if an attacker can establish a link between the person's identity and the tags he is carrying, the tracking of objects becomes the tracking of a person. This issue is mostly observed when the person moves to different places. However, in environments like home healthcare, this issue may not be of real problem, as the movement of the patient to be monitored is limited to a small area.

2.2.2 Symmetric Cryptography Primitives

Cryptosystems are divided into three main trends: symmetric or secret key, asymmetric or public key and unkeyed systems.

Asymmetric cryptography is built around mathematical hard problems [121] such as factorization, discrete logarithms or elliptic curves. Some approaches to adapt public-key cryptography for RFID, based on re-encryption and elliptic curve cryptography, are detailed in [20, 90]. Despite the efforts made to adapt asymmetric cryptography for low-cost devices,

such approaches are still considered beyond the capabilities of Gen2 tags [26]. The unkeyed cryptography main example is the hash function. The cost of these hash functions in hardware implementation is considered expensive in the literature [26, 112], due to the excessive consumption of hash functions regarding computational power and memory resources. Symmetric cryptography is based on the secret-key exchange between participants. There are two distinct types using symmetric encryption; stream ciphers and block ciphers. Their main functionalities are described as follows [141]:

- A block cipher is a function $E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$ which transforms an n-bit message block x into an n-bit string y under the control of k-bit key $k : y = E(k,x)$. The function is invertible in the sense that for each key the map $E_k \stackrel{def}{=} E(k,.)$ is a permutation of $\{0,1\}^n$, and the knowledge of k permits the computation of ${E_k}^{-1}$. In typical usage, a random key k is chosen and kept secret between a pair of users. x could be seen as an initial state that is initialized under the action of a key. The function E_k is then used by the two parties to process data in some way before they send it to each other.
- A stream cipher is a special case of block cipher where the transformation to a ciphertext is applied to a single bit (i.e., a block of one bit) for each cycle. It generates a keystream by sampling a constantly evolving state. The plaintext stream is then encrypted by combining it directly with the keystream to give the ciphertext stream. Encryption in stream ciphers is typically done using a bitwise XOR operation, known as *Vernam Cipher*.

The *Vernam Cipher* is the classical stream cipher example, defined by Equation (2.1), where m_i are the plaintext digits, k_i are the keystream digits and c_i are the ciphertext digits. The symbol $\oplus$ denotes the XOR operation (i.e., bitwise addition modulo 2), and the decryption is defined by $m_i = c_i \oplus k_i$.

$$c_i = m_i \oplus k_i \tag{2.1}$$

The *Vernam Cipher* is called the *one-time-pad*, if the keystream digits are generated independently and randomly. Thus, it is unconditionally secure against a ciphertext-only attack [164].

Passive RFID tags can at best be built using symmetric cryptography, based on key generation functions. The remainder of this section deepens into the keystream generators, called the pseudo-random number generators, which can be used by stream ciphers protocols.

2.3 Pseudo-Random Number Generators

Random numbers are widely used in either symmetric or asymmetric cryptography. As specified in the EPC Gen2 standard and ISO standards (ISO/IEC 18000-6C) [4], the usage of on-tag PRNGs on low-cost RFID devices is mandatory. Generally, random or pseudo-random numbers are used to generate keys, or nonces to be combined with passwords. Before delving into the details of PRNGs, we first look at true random numbers. True random numbers are totally non-deterministic. They can be produced by some unpredictable phenomena such as thermal and electronic noise of semiconductor. However, generating random numbers this way, is not practical for software applications because of its time-consuming. It also does not meet the requirements of certain class of protocols, where the same key should be shared by the two participants. Pseudo-random numbers are meant to appear as random even if they are a function of a deterministic machine. PRNGs are algorithms, which are initialized with a seed to produce a much longer sequence that appears to be random. Given the same seed value a PRNG will always produce the same sequence. The objective of PRNGs is to produce an independently random sequence of evenly distributed numbers that an adversary will have a low probability of prediction [121]. In the EPC Gen2 standard, the PRNG main function is to generate 16-bit sequences to be used as one time pad nonces for data encryption.

2.3.1 Main Properties

Common PRNGs consist of two components: (1) a generation function that, taking an internal state, generates the next output and then updates the internal state accordingly; and (2) a seed generation function that generates the initial state (and/or key) of the system. There are many nuances of PRNGs used in practice, which are often more complicated. For example, some of them are associated with auxiliary inputs such as time-stamps or counters, which also can be controlled by the adversary. PRNGs play an important role when encrypting plaintext messages in constrained environments. They are generally fast and have a simple hardware circuitry, solving the absence of buffering and being suitable for highly probable error-transmission channels [121].

There have been numerous works on constructing PRNGs for symmetric encryption schemes [18, 23, 54]. They could be based on a wide range of cryptographic primitives. The PRNGs that are in prelevant use today, are typically based on hash function or block cipher designs. Given the limited computing power of low-cost RFID systems (e.g., Gen2 tags), block cipher based implementation are most likely used. More information about block ciphers is given in Appendix A. In our work, a PRNG is defined as a pseudo-random bit generator whose output is partitioned into blocks of a given length n. Thus, the security of our proposal turns to secure the embedded block cipher. Section 3.4.1 of the next chapter details our solution regarding the PRNG. Some designers propose a model that combines the internal state and the key of the PRNG (cf. [18] and [23]), while others separate them [54].

The use of one of the two models depends on the primitives used to draw the PRNG. For example, the integration of Solitaire keystream as a PRNG in our model, imposes to combine the master key and the internal state only in the initial session. Nevertheless, the internal state has always an ephemeral nature, since it is usually updated during every iteration of the generation algorithm, whereas, the master key, which is also an initial state, has a much longer lifetime. Next, a common PRNG, named the Linear Feedback Shift Register (LFSR), is introduced, followed by relevant proposed PRNGs for low-cost RFID.

Linear Feedback Shift Registers

A linear feedback shift register (LFSR) is a digital circuit of n bits (or cells) that contains a shift register (or state) and a feedback function. The shift register is composed of a sequence of binary cells that share the same clock signal. Each time a bit is needed, the content of the register is shifted one cell, obtaining the most significant bit of the register in the previous state. The LFSR can be determined by the feedback polynomial function of degree n. This function is defined by the values of the coefficients $c_i \in \{0, 1\}$. If $c_i = 1$, the content of the cell $n - i$ is used to compute the new most significant bit of the register. The content of taps is processed with an XOR logical operation. Thus, the generated sequences of the LFSR depend on the polynomial function and the initial state of the register cells.

LFSR remains the most common shift registers-PRNG used in cryptography, due to its efficient and simple hardware implementations. But still, it has a significant drawback related to the predictability of its outputs (thus, initial states) [170]. In addition, with only 2^n output values, the Berlekam-Massy algorithm [115] can solve the feedback coefficients of an LFSR. As a result and because of the LFSR linearity problem leading to a predictable outputs (i.e., the linear complexity of a sequence is defined as the shortest number of LFSR cells that can generate the sequence), an LFSR should never be used alone as a keystream generator. To break the linearity with simple designs (to meet low-cost RFID devices), some approaches are introduced in the literature. We note mainly two of them [163]: the use of a non-linear filtering function as an input to the register, as shown in the Mifare PRNG [70]. This kind of generators has shown a vulnerability when the non-linear function is not taken carefully, for example, when the LFSR is always initialized with the same state values. Another approach to bypass the linearity of an LFSR is to use a combination of multiple LFSRs to generate a unique output. Examples of this approach are the A5/1, the first proposition for GSM standard and the Shrinking generator [160]. The output generated from these constructions is reported as statistically weak, as they are vulnerable to correlation and side channel attacks [87]. In addition, some techniques in [118, 122] provide that the feedback polynomials could be used to attack the scheme. In the sequel, we focus on some PRNG in the literature specifically designed for constrained devices, such as the EPC Gen2, we are interested in.

2.3.2 Related Work on PRNGs for Constrained Devices

Che et al. present in [35] a new PRNG intended for EPC Gen2 applications (following the randomness requirements stated in the standard). They propose an LFSR in which the linearity is perturbed by a physical source of randomness (a truly random bit) added for each cycle ring.

This combination leads to a generator with low consumption and hardware complexity. Nevertheless, in [119], a problem of non-randomness due to the inherent linearity of linear feedback shift registers is shown. In addition, the obtained PRNG cannot be used as a part of a standard communication model, where the sender and receiver should reproduce the same sequence at both sides. This is due to the cipher sequence affected by a true source of randomness.

FERNS [51] is another proposal to derive truly random data using the initial state of tag Static Random Access Memory (SRAM). The method shows that the initialization of the SRAM produces a physical fingerprint. Thus, the authors propose a system of Fingerprint Extraction and Random Numbers in SRAM (FERNS) that harvests static identity and randomness from existing volatile memory storage. Experiments show that extracting 128-bit patterns from a testing device with 2,048-bit of memory fails to pass entropy tests [51]. The authors propose the use of Universal Hash Functions to improve the output randomness, which is beyond the capabilities of low-cost devices. Despite of these results, the FERNS method remains interesting and power efficient as TRNG, which could be rather useful in combination with other PRNG models. These models are different from those that interest us in this dissertation.

Grain [113] is a proposal from the eSTREAM project [58], designed for hardware efficient implementation. It is based on the combination of two feedback shift registers. The first uses Linear Feedback Shift Registers (LFSR) while the second uses a Non-Linear LFSR (NFSR). However, because of the NFSR and the fact that its input is masked with the output of the LFSR, the exact period of this PRNG depends on the key and the used initialization state [163].

LAMED-EPC [145] is a low-cost PRNG specifically designed for EPC Gen2 devices. Originally, LAMED builds a 32-bit PRNG. However, as EPC Gen2 requires the use of a 16-bit PRNGs, the authors propose an alternative 16-bit version of their PRNG. To reduce the output from 32 to 16 bits, the authors divide the first output in two halves, namely $MSB_{31:16}$ and $LSB_{15:0}$. Then, an additional XOR operation is done on the two halves, before the final output sequence. The LAMED-EPC requires some more circuitry and clock cycle than the first version, nevertheless, an exhaustive statistical analysis confirms that both proposals successfully satisfy the initial expectations. Apart from hardware complexity and statistical behavior, there is no evidences of further achievements. The inherent particularity of their

construct methodology (adaptation of a first proposed scheme), makes it difficult to compare with other designs in the literature.

So far, we particularly focus on designs of PRNG, which are used as a part of a communication protocol. Once generated, these keys are meant to be subsequently used for message exchange between parties involved in a communication protocol. This is the aim of the next section, where an overview and related work on the way these keys are exchanged is provided.

2.4 Key Establishment Protocols

Key establishment protocols could be used to rekey the internal system states (e.g., states of a PRNG) in order to ensure that only fresh keys are available in new sessions. This section considers key establishment protocols that provide shared secrets between two parties, typically for subsequent use as symmetric keys for a variety of cryptographic purposes including encryption and entity authentication.

2.4.1 Key Establishment Techniques

A protocol is a multi-party algorithm, defined by a sequence of steps. These steps precisely specify the actions required by two (or more) parties, to achieve a specified objective. Key establishment is a protocol which aim, is to make available a shared secret to two (or more) parties, for subsequent cryptographic use [121]. Key establishment may be broadly divided into *key transport* and *key agreement*.

A *key transport* mechanism is a key establishment technique where one party creates (otherwise obtains) a secret value, and securely transfers it to the other(s). On the other hand, a *key agreement* mechanism is a key establishment technique in which a shared secret is derived by two (or more) parties as a function of information contributed by each of these, such that (ideally) no party can predetermine the resulting value. The most known algorithm related to the *key agreement* mechanism is the Diffie-Helman and its variations [180]. Additional variations beyonds *key transport* and *key agreement* exist. They include various forms of *key update*, such as *key derivation*.

Key establishment protocols involving an authentication step, typically require a set-up phase, where authentic initial key is distributed. Most protocols have as objective the creation of distinct keys with each protocol execution. In some cases, the initial key is pre-defined and set every time the protocol is executed. Systems involving such static keys are insecure under known-key attacks.

Regarding EPC Gen2 communications, the use of *key agreement* techniques may require heavy computation towards the Gen2 tag [96]. Indeed, these techniques expect equivalent capacity between the reader and the tag. Since RFID readers are generally designed to

have enough computational resources to calculate robust keys, we argue that a *key transport* technique may suit protocols involving Gen2 tags. The *key update* can also be achieved by *key transport* or *key derivation*. For *key update*, the derived keys are based on a per-session random input provided by one party (cf. protocols in [121]). Although this technique refreshes keys in every new session, it is still vulnerable to forward secrecy [121] as the compromising of the established long-term keys does compromise all past session keys. Thus, additional effort regarding the refreshment of the long-term keys is to be considered. This constitutes one of the challenges that our proposed protocol in Chapter 3 should meet.

The remainder of this section presents a number of key establishment protocols for authentication goals in the literature, involving RFID tag communication.

2.4.2 Related Work on Key Extablishment Protocols

Based on the cryptographic techniques seen above, key establishment protocols provide shared secrets between the communication entities. These secrets are meant to be subsequently used for different purposes, typically, for entities authentication. For example, in EPC Gen2 protocols, a successful authentication after a successful key exchange, authorizes the reader to access the tag additional memory. There have been several attempts to create authentication protocols for RFID systems. These attempts are divided into four major trends: hash based, Physically Unclonable Functions based, lightweight, and bitwise based solutions.

Hash based solutions

Most of the first security methods based their design on the use of one way hash functions. For instance, in [182], the authors proposed a hash-lock function to prevent unauthorized readers from reading tag contents. The design principle behind this method includes the assumption that tags cannot be trusted to store long-term secrets when left in isolation. Thus, the secret is first sent by authorized readers to tags using a trusted environment. Then, equipped with an internal hash function, the tags perform a hash on this secret and store it within their internal memory. Then, tags enter into a locked state in which they answer to any possible query with the computed hash. Some weaknesses of the hash-lock scheme were shown in [79] leaving the protocol vulnerable to eavesdropping and replay attacks.

Another set of hash-based solutions were proposed in [136], [74], [79] and [80]. In [136], the authors propose the use of hash chains to implement on-tag security mechanism requiring the tag to perform a hash operation on its ID and to send the result to the reader. The reader then makes an exhaustive search for a match. The limitations of this solution were discussed in [74] (e.g., the protocol is subject to replay attacks). The authors propose an enhanced hash-based RFID protocol to address privacy and authentication by introducing a time variable to prevent replay attacks. The solution in [79] follows the same vein, in that the reader and tag

compare hash results. A more general solution, based on hash chains, is proposed in [177] to guarantee mutual authentication between tags and readers. The use of chains is reported by the author as very memory consuming and the solution is qualified as prone to availability attacks on the server side. Moreover, since the publications in [112, 26], most authors in the literature agree that the use of hash functions is beyond current capabilities of low-cost RFID devices, such as Gen2 tags.

Physically Unclonable Functions based solutions

Physically Unclonable Functions (PUFs) are often used as a complement to hash functions in more powerful RFID solutions [93]. Half way between traditional cryptography and physical protection defences, the PUF technology has traditionally been used to implement challenge-response protocols. PUFs originated in [143] with the conception of optical mechanisms for the construction of physical one-way functions. The key idea is to fingerprint tags, based on their physical (i.e., fabrication) properties, e.g., delay properties of the tag circuitry or startup values of the tag memory. The main drawback is the necessity of a great number of challenges (i.e., hundreds of them) exchanged between readers and tags at a given time, to conclude the authentication process. Moreover, as it happens with many other probabilistic identification schemes [106, 149], the solution may expose the identification process between peers to an increase in response delay and power consumption, and might be acceptable only on short-distance RFID technologies with their radio spectrum in low (LF) and high (HF) frequency bands.

The authors in [27] propose a solution to complement the use of PUFs with message authentication codes, aiming to simplify the challenge-response communication scheme of previous proposals. The approach does still require the necessity of huge lists of challenge-response pairs for each PUF/tag which must be stored on back-end servers connected to the readers. Indeed, once a given pair is sent, it must not be used anymore. Otherwise, the protocol cannot guarantee that an adversary eavesdropping data will not gain advantage by performing a replay attack. Distance bounding schemes, used in protocols like [140, 129], are meant to counter this kind of attacks by handling extra delays in the exchange of messages. However, we are unaware of distance bounding protocols for UHF Gen2 RFID technologies, given their delay and power constraints.

Lightweight solutions

With the motivation that traditional cryptographic protocols are too computationally intensive to be utilized, the authors in [91] presented the HB+ protocol which was inherited from an early work [81]. This improved variant attempts to prevent active attackers using a statistical conjecture to bound the difficulty of learning a secret (e.g., the ID of the tag) given a sequence of randomly chosen vectors with embedded noisy information. Thus, the security

relies to the difficulty of the Learning Parity with Noise (LPN) problem, which has been proved to be NP-hard. The success of the man-in-the-middle attack in [73] on HB+ leads to other versions following the basic HB+ protocol, we cite HB++ [30], HB* [56], Modified HB+ [150], HB-MP [131], Random-HB# & HB# [72] and Trusted-HB [29].

Random-HB# & HB# have not been reported attacked yet, but the use of Toeplitz Matrix for HB# and the use of an LFSR-based Toeplitz Matrix to construct a hash function for Trusted-HB are rather expensive for constrained tags like those of Gen2 technology. HB-MP protocol was the best candidate suited to Gen2 tags. However, it was explained in [72] that the amount of data transferred remains high. Furthermore, in [157], the authors report that the protocol is vulnerable to man-in-the-middle attacks. The vulnerability found on HB-MP is due to the predictable repetition of the internal state in each session. In [103], the authors propose a solution to randomise the internal state, relying on some hash function. The solution, seems to be not yet supported by basic Gen2 tags until now. Finally, HB-PUF is a PUF version of the HB series [77]. Its main drawback is once again the requirement of storing a very large number of challenge-response pairs at either the reader or the back-end database.

Bitwise based solutions

Several other approaches aim at providing authentication based only on bitwise operations and very low computational primitives. The MAP family of protocols by Peris-Lopez et al. is an appropriate set of protocols characterising this class of lightweight protocols. We note LMAP [147], M2AP [148] and EMAP [146]. Their goal is to provide mutual authentication between readers and tags. The authors eliminate the use of hash primitives and involve only modular arithmetic on-tag operations. The computation of costly operations is done at the reader side. Although this effort to lighten the implementation in the tag side, none of these proposal seems to resist adversaries able to learn the secrets (including the identifier) of the tag with relative ease [104, 124, 125, 123]. For example, M2AP was broken by eavesdropping for two consecutive runs of the protocol. Improvements of these schemes [38] have also been reported as vulnerable [33]. All these examples show the necessity of formally verifying the security of new authentication schemes before their release.

Finally, other recently efforts in [179, 32, 78, 31, 95, 16] have analyzed forward security and other communication faults in RFID systems at various levels of formality. These works are the closest to our proposal (i.e., authors propose a protocol and a formal approach to verify its security). In Section 4.3, we analyze them more in detail and provide a comparison with our work.

2.5 Verification Approaches

The process of verifying cryptographic protocols has led to the development of two approaches, namely: the computational (or the cryptographic) approach and the symbolic (or the formal) approach. Each approach relies on mathematical models for the executions of primitives/protocols in adversarial environments. More specifically, each one formally defines security properties expected from cryptographic systems, and develops methods for rigorously proving that the given system constructions meet these requirements.

The computational approach considers issues of complexity and probability of the bit-strings messages where the cryptographic primitives are functions from bit-strings to bit-strings. This approach guarantees security against all probabilistic polynomial-time attacks. It is generally admitted that security proofs in this model offer powerful security guarantees. A serious drawback of this approach is that proofs for even moderately-sized protocols are usually long, difficult, and highly error prone [47]. It complicates the proofs, and until recently, these proofs were only manual.

In contrast, the symbolic approach uses a highly abstract view of the execution, where the messages exchanged by parties are symbolic terms and the adversary is restricted to only use these primitives for computations. Besides, in this approach, primitives are assumed to be absolutely secure. The resulting models are considerably simpler than those of the computational approach. Therefore, the proofs are also simpler, and can benefit from the support of machines. In fact, the symbolic model, is more suitable for automation, essentially by computing the set of all messages the adversary learns. Starting in the 1990s, the proof of protocols in the symbolic approach has been an important application field for formal verification methods [25]. The remainder of this section deepens into the symbolic approach.

2.5.1 Symbolic Approach

In the symbolic approach, the formal methods use a language that combine the modeling of a cryptographic protocol and its security properties, together with an efficient procedure to determine whether a model does indeed satisfy those properties requirements [117]. Specification language could be a formal language or calculi. Three types of protocol analysis are shown [169]: *inference construction*, *attack construction*, and *proof construction*:

- *Inference construction* techniques are based on modal logics. The basic idea of these approaches is the analysis of knowledge or belief during the protocol run. Inference rules are used to describe how knowledge can be derived from other knowledges, to show if certain conditions are satisfied.
- *Attack construction* techniques attempt to discover vulnerabilities using algebraic properties of the algorithms of a protocol. The majority of the *attack construction* techniques

make a form of a state space exploration. Thus, a model of the system is defined, then analyzed in an attempt to discover a path to an insecure state.

- *Proof construction* techniques model the computations performed in a protocol and define the security properties as theorems. Automated theorem checkers are then used to verify these theorems and thus prove the properties of the model.

In [116], authors argue that *inference construction* techniques are in general weaker than *attack construction* because they operate at a higher level of abstraction. Hence, interest in *inference construction* has decreased, while *attack construction* methodologies have improved. On the other hand, it is claimed that *attack construction* and *proof construction* techniques are complementary [144]: *attack construction* techniques are typically easy to use and can provide an assurance that a model satisfies the security criteria. *Proof construction* techniques are more complicated, but allow a more thorough analysis. Before dealing with existing verification tools, we detail in the sequel, the Dolev-Yao adversary that is typically used in symbolic approaches.

Dolev-Yao adversary. In this model [52], the adversary has a complete control of the network: messages may be read, modified, deleted, or injected. The adversary is also able to manipulate data contained in those messages, under the restriction of perfect cryptography. Thus, the attacker is only able to perform cryptographic operations when possesses the necessary keys. For example, an adversary, given the required keys, may decrypt ciphertexts. The relationships between cryptographic primitives are captured by a set of deduction rules. *Attack construction* methods typically assume that cryptographic protocols should achieve their objectives in the presence of the Dolev-Yao adversary that has full control of the network.

Formal methods have proven to be a promising technique towards developing automated and generic protocol verification and testing methods. *Attack construction* is a particular method that shows an improvement in automated verification. The most significant problem of this kind of methods is that the search space of the paths to find the insecure state can be infinite, which leads to a termination problem. In the sequel, we consider these automated techniques, which allow the formulation of precise security statements about cryptographic protocols and the conditions under which these statements hold.

2.5.2 Automated Verification Techniques

The inability of experts to identify the flaw in the Needham-Shroeder [133] public key protocol highlights inherent weaknesses in the manual verification process and necessitates automated analysis tools. A model checking is one of the most common formal verification techniques in the symbolic approach. Typically, it is a method commonly used for the automatic verification of reachability properties. Given a system and a property p, reachability model checking is

based on an exhaustive exploration of the reachable state space of the system, testing whether or not there exists a state where p holds.

State space exploration problem Although the automatic verification of protocols in the symbolic approach is easier than in the computational approach, it still presents significant challenges, essentially related to the infinity of state space exploration. This is due to two reasons: messages are of arbitrary length, and the number of sessions (i.e., protocol runs) is unbounded. Regarding the number of participants, we can easily bound it to the protocol without ignoring attacks [45]: for protocols that do not make difference tests, one honest participant suffices for the secrecy property if the same participant is allowed to play all roles of the protocol, whereas two honest participants are sufficient for the authentication property.

State space exploration approaches One solution to the problem of infinite states is to explore only part of the state space, by limiting arbitrarily both the message length and the number of sessions of the protocol. Some standard model-checking techniques use this approach, such as FDR [107] (which was used to discover the known attack against the Needham- Schroeder public-key protocol), Maude [53], Murϕ [128], and SATMC (SAT-based Model-Checker) [14]. These techniques can find attacks against protocols, but do not prove the absence of attacks, since attacks may appear in an unexplored part of the state space. In [108], the author demonstrated that the limitations of finite state space can be overcome by introducing manual proofs.

More generally, an arbitrary number of sessions is a reasonable choice [45]. If only the number of sessions is bounded (i.e., the number and length of messages sent by the adversary remain unbound), the verification of protocols remains decidable, meaning that the protocol insecurity is NP-complete with reasonable assumptions on the cryptographic primitives [155]. The verification is much more difficult, when cryptographic primitives have algebraic relations, but the complexity class does not necessarily increase. For example, the XOR operator is handled in [37, 46], in the case of a bounded number of sessions, still with an NP-complexity.

Practical algorithms have been implemented to verify protocols with only a bounded number of sessions. Some implementations use constraint solving, such as the work in [126] and CL-AtSe (Constraint-Logic-based Attack Searcher) [178], and others use extensions of model checking such as OFMC (On-the-Fly Model-Checker) [19]. Both CL-AtSe and OFMC support the properties of XOR operator. They are part of the AVISPA (Automated Validation of Internet Security Protocols and Applications) plateform [13]. For an unbounded number of sessions, the problem is undecidable [132] for a reasonable model of protocols. Despite this undecidability, many techniques have been designed to verify protocols with an unbounded number of sessions. For example, they use restrictions to subclasses of protocols, or require user interaction, or tolerate non-termination or incomplete systems (i.e., the result remains unknown). Most of these techniques deal with trace properties. An exception are the system of Abadi [5] and ProVerif [100] which deal with equivalence properties, called also

indistinguishability. ProVerif provides support for equivalence in addition to reacheability, however, due to over-approximation, it may report false attacks (cf. More details on protocol verification techniques are recently surveyed in [25]).

In conclusion, each tool has some strengths and weaknesses. Apart from the verification tool, the choice depends on the verification aim: whether the verifier wants to find vulnerabilities in the protocol (*attack construction* technique) or to search for proving the correctness and completeness of the protocol (*proof or theorem construction* technique). It also depends on the intention to use automatic tool with a manuel aided proof approach (i.e., when exploring limited state space) or to entirely rely on the output of the automatic tool. In our case, we rely on the CL-AtSe model checking tool dealing with the *attack construction* technique. The aim is to verify the robustness of a key exchange protocol against the Dolev-Yao adversary. Although the advantage of the symbolic analysis, this latter might miss attacks that exploit weaknesses of cryptographic primitives. We show the weaknesses of the PRNG which is used as cryptographic primitive in our protocol and provide a mean to evaluate it. For this aim, we use a statistical tool to verify the randomness of the PRNG outputs, cf. Chapter 4.

Conclusion

We have introduced the main characteristics of the EPC Gen2 standard for low-cost passive RFID systems. We have seen that the communication protocol between readers and tags should be simple and robust enough to protect critical operations such as memory access and tag killing. Besides, we have seen that PRNGs are used as keystream generators for Gen2 tags, due to their suitable properties for hardware implementation in constrained environments and for their good statistical properties. However, Gen2 tags suffer from a flawed security protocol regarding the execution of critical operations on tags. In addition, more effort regarding the PRNG should be realized to enlarge the period of key predictability. A number of key establishment protocols have been presented as a solution to passive RFID tags. Some of them proposed models and cryptographic primitives beyond the capabilities of the Gen2 tags, while others presented lightweight solutions but, fail to achieve long term security of the exchanged messages. This can be caused by the PRNG implemented in the tag or by the communication protocol that is designed in a bad way. To end this chapter, main formal verification approaches are presented, providing their advantages and limitations. A focus on the model checking tools has been provided to ease the verification of security protocols.

Motivated by the limitations of EPC Gen2 security model and the need of proved and validated security protocols, we present in next chapters, a communication protocol designed to share and refresh security keys between readers and passive RFID tags. Furthermore, a symbolic verification approach of the proposed protocol is given to verify strong notions of security. Finally, a proposition of a lightweight PRNG is also presented, and some statistical evaluations are given.

CHAPTER 3

KEDGEN2: A Key Establishment and Derivation Protocol

"Be you and your soul will lead you to the right path." – Mustafa Mahmoud

Introduction

Passive Radio Frequency IDentification (RFID) tags are constrained devices where introducing security is an important and challenging task. The Electronic Product Code Generation 2 (EPC Gen2) standard is a RFID based system, used to assign a globally unique number to every RFID tag. It also stores tag related information in its additional memory, which could include secret keys and bearer's related information. Although its wide use, the EPC Gen2 technology still presents some important flaws , e.g., the password exchanged to access the tag memory [89]. Indeed, obtaining the password to access the memory of an EPC Gen2 tag permits to a malicious adversary to access and modify the memory of the tag. When the memory of the tag includes additional information about objects linked to people holding them, information disclosure can be an important threat. Such a threat can be handled by modifying the security model of the Gen2 protocol specification and providing a more elaborated way of managing cryptographic keys in Gen2 systems. This is the aim of the present chapter. We propose a communication protocol designed to share and refresh keys between readers and tags. In addition, we propose a pseudo-random number generator to update keys for every message. First, we detail the system assumptions where our protocol, named KEDGEN2 is to be applied. Then, we deal with the targeted security properties, under which our protocol will be verified. Next, we provide the protocol specification, its modeled key generation function and its components. The following section introduces the Solitaire keystream generator as a solution to generate keys in each exchange between reader and tag. This generator should meet the requirements of passive RFID tags and permit to preserve secure keys as long as possible. To ensure long term secure communication, we detail the cycle length detection of Solitaire keys before updating its initial state.

3.1 System Assumptions

3.1.1 Revised Security Model

The security model in EPC Gen2 systems relies on the communication of one-time sequences used to encrypt sensitive data that must be sent over the insecure RFID channel. The purpose of generating one-time sequences for security is typically that both entities participating on the communications are able to repeat the sequence. However, this is not the case in the EPC Gen2 protocol, where only the tags have access to the sequence generation function. Therefore, the generated sequences cannot be reconstructed at the reader side, and must be sent as clear text over the insecure channel (i.e., tag-to-reader channel) [71]. To handle this security flaw, we propose to use an alternative model, and assume the use of a pseudo-random number generator (PRNG) whose algorithm and internal states are known at both readers and tags sides.

3.1.2 Adversary Model

We assume an active adversary $\mathcal{A}$ who controls the communication channel shared between tags and readers. Therefore, $\mathcal{A}$ can eavesdrop, store, analyse, alter, redirect, and reuse intercepted messages. $\mathcal{A}$ always knows the non-secret data and the functions that each part execute, as well as the inner working of the system (e.g., algorithms and environment associated with the protocol). Additionally, $\mathcal{A}$ can *impersonate* a reader or a tag, and inject new messages by such controlled entities. However, $\mathcal{A}$ cannot modify those messages already sent by a non-controlled entity, nor can $\mathcal{A}$ prevent non-controlled entities from receiving a message already sent. Finally, $\mathcal{A}$ is motivated by any possible scenario leading to the disclosure of secret information used in the protocol. Therefore, we expect from $\mathcal{A}$ the application of the following scenarios:

- *Protocol exposure.* $\mathcal{A}$ can try to find any protocol flaw to decrypt the derived keys relying on its *a priori knowledge of the system*. Therefore, $\mathcal{A}$ can try to find any link between captured messages to correlate two or more protocol outputs. The aim is to obtain information about the derived keys.
- *Key recovering.* Using the derived keys, $\mathcal{A}$ can try to detect, at least, a couple of internal state and derived key to elaborate a relation between them. The aim is to conduct an exhaustive key search attack (cf. Appendix A) to derive the master key.

3.2 Targeted Security Properties

The protocol shall provide secrecy of the master and derived keys in addition to assuring that mutual authentication is done between honest participants preventing impersonation attacks. Strong notions of secrecy such as forward and backward secrecy must be also verified even if adversary $\mathcal{A}$ corrupts the whole system by obtaining the session master key and the internal state of the key generation function by external means (e.g., by physically exposing the data of the tags). Therefore, our protocol should guarantee the security properties defined below.

3.2.1 Mutual Authentication

We define mutual authentication by the agreement of the reader and the tag on the value of a negotiated master key in each session. When this key is also proved to be secret (i.e., nobody except the intended parties knows the key), this strong agreement excludes potential man-in-the-middle and replay attacks in which the adversary could impersonate one of the two parties.

3.2.2 Secrecy of the Master Key

At any time period, $\mathcal{A}$ cannot recover the master key from the derived keys used in a given session and within the valid period of generation (i.e., before reaching a given threshold N).

3.2.3 Forward Secrecy

After the exposure of a given master key, $\mathcal{A}$ cannot compute previous master keys used in the system once the master key is refreshed. In other words, let Km_i be the i^{th} master key negotiated between the tag and the reader, t_i be the last instant of the time interval during which Km_i is in use, and t_C be the instant of the total compromise event of the tag and the reader. That is, when $\mathcal{K}_t$ is the knowledge of $\mathcal{A}$ at instant t after t_C, then $\mathcal{A}$ cannot deduce Km_i from $\mathcal{K}_t$: $\mathcal{K}_t \nvdash Km_i \ ; \ t_i < t_C < t$.

3.2.4 Backward Secrecy

After the exposure of a given master key, $\mathcal{A}$ cannot compute future master keys used in the system after the master key is refreshed. In other words, let Km_i be the i^{th} master key negotiated between the tag and the reader, t_i be the first instant of the time interval during which Km_i is in use, and t_C be the instant of the total compromise event of the tag and the reader. That is, when $\mathcal{K}_t$ is the knowledge of $\mathcal{A}$ at instant t after t_C then, $\mathcal{A}$ cannot deduce Km_i from $\mathcal{K}_t$: $\mathcal{K}_t \nvdash Km_i \ ; \ t_C < t_i < t$.

3.3 The KEDGEN2 Protocol

Our protocol assumes dynamic master key establishment based on key transportation techniques [121]. This rationale is used since parties in our system have not the same capabilities. Indeed, RFID readers are expected to have enough computational resources to calculate robust keys. Once computed, the master keys are communicated to the tags, assumed to be resource constrained components. We first present our key generation model. Then, we detail the protocol that uses these generated keys to encrypt secret data.

3.3.1 Modeling the Key Generation Function

Let $\mathcal{BS} = (\mathcal{K}d, \mathcal{E}n, \mathcal{D}c)$ be the base symmetric encryption scheme, specified by its key generation $\mathcal{K}d$, encryption $\mathcal{E}n$ and decryption $\mathcal{D}c$ algorithms. Let $\mathcal{G}en = (\mathcal{S}, \mathcal{G})$ be the PRNG based on a block cipher primitive whose block size is the length of the derived key of the base scheme. $\mathcal{G}en$ consists of two algorithms. The first algorithm $\mathcal{S}$ is a probabilistic algorithm which takes no inputs and outputs an initial state St_1 and a master key Km_1. The second is an iterative deterministic generation algorithm $\mathcal{G}$, computing in each iteration an output Kd and a new state St_i. The counter avoids cases where the same state and key are used. It is a replay defense. For $i \geqslant 1$, the generation algorithm $\mathcal{G}$ takes as input the key Km and/or the current state St_{i-1} (including the cnt) to generate Kd_i and St_i as: $Kd_i, St_i \leftarrow \mathcal{G}(Km, St_{i-1})$. We associate with our PRNG a re-keyed encryption scheme, which establishes a new key Km in every new session. The re-keying function can rely on a one way function that is responsible for changing the keys for each session.

An encryption/decryption process of the model we propose is pictured in Figure 3.1. The objective is to encrypt every secret message with a new derived key Kd using $\mathcal{E}n$. Thus, the derived keys are used once in each transaction while the master key Km has a longer life time.

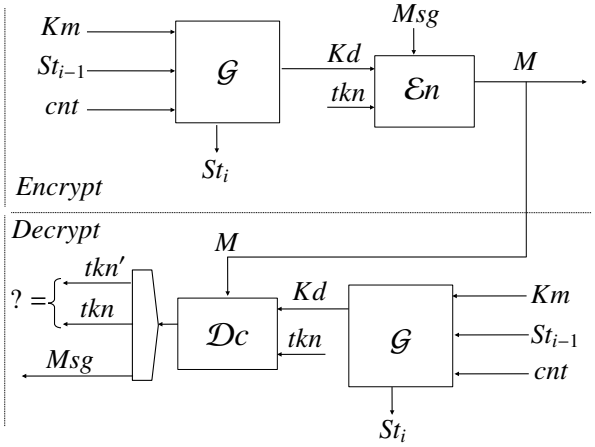


Figure 3.1 – Proposed encryption/decryption scheme

Notice that our aim is to minimize the advantage (i.e., the likelihood) of the adversary to compromise the security of $\mathcal{G}$ using the data he recovered in each transaction. The obvious attack that takes advantage of the weaknesses of the encryption under block ciphers is the birthday attack [21]. To safely encrypt more data, a practical solution is to enlarge the limited threshold leading to birthday attacks. For basic block ciphers solutions, cf. Chapter 2, it is possible to use the results of [6] by introducing a master key re-keying every $N = 2^{n/3}$ encryptions, where n is the block length. The solution increases the encryption threshold from $N = 2^{n/2}$ to $N \approx 2^{2n/3}$. This solution requires less resources than the data dependent re-keying (cf. Appendix A). In addition, it follows our protocol design in refreshing the keys every new session.

With this re-keying function, our encryption scheme is divided into several stages in a same session. In stage i, all encryptions are performed using the base scheme with Km_i. An encryption counter is maintained, and when N encryptions are performed, the stage ends and a new stage begins with a new counter *cnt* and a new master key.

3.3.2 Protocol Components

We assume the following components:

- *Tag.* A passive constrained device that communicates with readers via a radio interface. The tag is able to give access to its memory only with one reader at a time. It holds the function $\mathcal{G}$ of the generator $\mathcal{G}en$ and is able to derive keys according to its secret master key and state.
- *Reader.* An active entity communicating with the tags and a back-end server. It implements a radio interface to tags and a trusted interface to the server. It holds the functions $\mathcal{G}$ and $\mathcal{S}$ and is responsible for refreshing master keys when necessary.
- *Back-end server.* A trusted entity that stores in its database all tags and readers information. It is responsible for setting up the initial keys either in the tag or in the reader. It also operates to reset the system when problems arise.
- *Channels.* There are a reader-to-server channel and a reader-to-tag channel. The readers and the server are related with a high-level security channel. They are assumed to be secured with common security protocols (e.g., SSL/TLS). Reader-to-tag channel is the vulnerable channel that captures our interest.
- *Sessions.* We separate each execution of the protocol by a process named session. For each communication session between a pair of reader and tag, a different master key is established. Session key ensures the independence across sessions to avoid long-term storage of shared keys and to limit the number of ciphertexts available for cryptanalysis.

3.3.3 Protocol Description

In the remainder of this section, we describe the steps of the protocol. For the sake of simplicity, the reader and the server refer to one entity named *sensor*, since (i) readers do not store locally secret information related to tags, and (ii) the linking channels to the server are assumed to be secure.

Each tag and sensor store a generation algorithm $\mathcal{G}$ (cf. Section 3.3.1) and generate keys with a synchronized process. $\mathcal{G}$ is deterministic. Thus, given Km_i and $St_{i,j-1}$ in the i^{th} session and $(j-1)^{th}$ derivation, $\mathcal{G}$ always outputs the same derived key $Kd_{i,j}$ and State $St_{i,j}$. The function of initialization $\mathcal{S}$ is performed once, (i.e., the first time the protocol is executed). It can be re-called if the system has to be set up. The sensor stores in its database all the tag information. For each tag, it records the tag pseudonym (or identifier) TagID, its current state St_i, the master keys (Km_{i-1}, Km_i) to recover the last key in case of desynchronization, a generator counter *cnt* (cf. Section 3.3.1) and an encryption token *tkn*. The token *tkn* can be a counter or a timestamp. In our scheme, we propose the use of a counter since tags are not usually connected to a server that can synchronize their clocks. We assume in all transactions that *tkn* guarantees that the sent messages will be different from the ones sent in the previous transactions. It is meant to ensure the message integrity.

In case of loss in the transmission due to interference or noise, the messages are assumed to be resent with the same counters *cnt* and *tkn*. That is, if the reader or the tag does not receive the acknowledgement of the last message, the message can be retransmitted with a bit set to indicate that it is a duplicate. Hence, the receiver accepts only one validated message.

The master key Km is sent to the destination whenever the key generator $\mathcal{G}$ needs to be refreshed. This happens in the two following situations:

1. when a new session begins. In this case, the sent master key becomes the session key,
2. when the key generator counter *cnt* reaches a threshold N leading to the possible birthday attack. At this time, $\mathcal{G}$ has to be rekeyed with a new master key to extend its lifetime against the birthday attack. In this case, the sent master key replaces the actual session key.

In the sequel, we present the different steps of the protocol in more detail.

Sessions

A set-up phase is required for initializing the state and the master key Km. In this phase, authentic and secret initial keying material is distributed by a trusted third party over a secure channel.

First session. The tag and the sensor agree on an initial secret composed of: an initial master key Km, an initial internal state *seed* and a shared token *tkn*. The *cnt* of the $\mathcal{G}$ function is also initialized.

i^{th} *session.* In the beginning of the i^{th} session (i.e., before refreshment), the sensor and the tag share the function $\mathcal{G}$ with the same properties as those used in the $i-1^{th}$ session, meaning that they use Km_{i-1} for generating derived keys. The period of generation is assumed to be still valid for unpredictable derived keys. After the establishment of the master key, the tag and the sensor share: (1) a secret master key Km_i, (2) an internal state St_i associated with a new counter *cnt* and (3) a token tkn_i, which is newly refreshed.

Refreshing $\mathcal{G}$ in the same session

When the generation counter *cnt* reaches the value of N, the sensor sends a query for refreshing $\mathcal{G}$ as follows: Sensor sends to the tag a special command $Scmd$ xored with a new derived key $Kd_{i,j}$. Tag acknowledges the refreshment with $Kd_{i,j+1}$. Then, Sensor sends a new master key. The tag verifies the derived key and the token used to encrypt the message containing the master key and in case they are equal to those it has already calculated, it accepts the master key.

Stages

In each new session, three stages are required. One stage for identification and another for authentication and key establishment. The third stage is a consequence of the successful

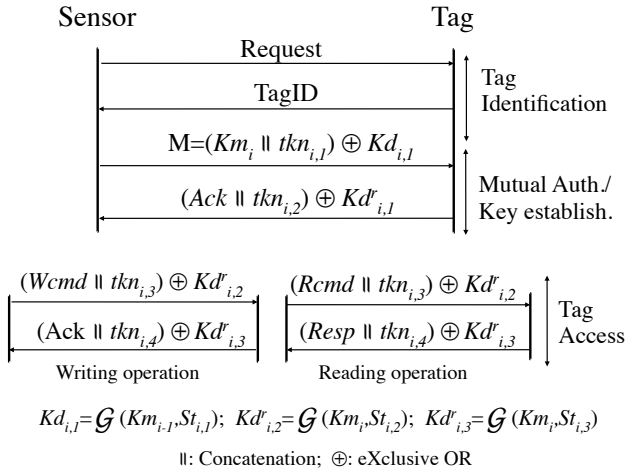


Figure 3.2 – Main stages of the protocol

authentication which means the access to the internal memory of the tag. These stages are shown in Figure 3.2.

1. Tag identification stage. The sensor starts by sending successive requests to the tag until it obtains the tag's pseudonym TagID. The sensor checks in its database the received TagID. If there is a match, the sensor associates the TagID with the EPC identification and the related secret information (i.e., the master key and the state previously negotiated). Note that for reasons of privacy, the EPC code is replaced by a pseudonym TagID. Both sides must have the same secrets. Otherwise, the next authentication process will fail. The sensor calculates its response including the derived key (by using $\mathcal{G}$ with the valid previous master key Km_{i-1} and St_{i-1}) to prove that it recognises the tag and *xor*es the result with the new established master key Km_i. Notice that the sensor stores both the current and previous master keys to handle desynchronization.

2. Mutual authentication and refreshment stage. Upon receiving the message M, Tag checks the derived key used for encryption. Then, it calculates a new $Kd'_{i,1}$ (i for current session) and decrypts M by applying an *xor* operation as follow: $Op = (Km_i, tkn_{i,1}) \oplus Kd_{i,1} \oplus Kd'_{i,1}$. If the decrypted suffix of Op is equal to the predefined token $tkn_{i,1}$, then Tag authenticates Sensor and accepts Km_i as a new master key. It returns an acknowledgement Ack associated with a new derived key $Kd^r_{i,1}$ (r for refreshed key) set from the refreshed values. Otherwise, Tag does not accept the sensor's key and aborts the communication. Upon receiving the value of $Kd^r_{i,1}$, Sensor verifies it and authenticates Tag, in case of validity.

3. Tag access stage. After a successful authentication, the sensor is authorized to access the tag. Thus, it has the ability to execute privileged commands like reading or writing on it. The same process of authentication is used to perform an access operation. Instead of sending the master key, the sensor sends the data to be written on the tag or the tag sends the data required by the sensor encrypted with a fresh derived key:

- Writing operation: Sensor begins by sending the write command $Wcmd$ concatenated with the token and *xor*ed with a new $Kd^r_{i,2}$. Tag verifies this key and accepts the command if the value is valid. Then, it acknowledges the reception with $Kd^r_{i,3}$. Otherwise, Tag aborts the communication.
- Reading operation: Sensor begins by sending the read command $Rcmd$ *xor*ed with the new generated key $Kd^r_{i,2}$. If Tag accepts the request by checking $Kd^r_{i,2}$, it sends the response $Resp$ *xor*ed with $Kd^r_{i,3}$.

Concurrent executions

The Gen2 technology is highly concurrent as a large number of tags could be interrogated at the same time. Thus, it is important for participants to separate concurrent protocol

executions. This issue is usually handled by adding a session ID field to the exchanged messages. In our protocol, we assume that each protocol session is associated with an initial internal state, a secret Km, and a token tkn that differentiate all the tags in the system. The sensor can run concurrently many protocol sessions at a time since it maintains a set of tag information. In contrast, the tag cannot run concurrently many protocol sessions at a time, particularly when it needs to update its secrets simultaneously (e.g., the secrets have to be updated before starting a new session). In the sequel, we consider that the tag can respond to several identification requests by sending its pseudonym but for privileged requests (reading/writing accesses), the tag does not respond to simultaneous queries, nor is it able to increment its internal state and token two times simultaneously. Finally, RFID systems have to deal with potential risks of loss of communication against channel troubles or special attacks such as jamming and blocking [97]. More specifically, when the communication session does not end properly, the tag could not be available to respond to next sensors requests. Thus a desynchronization problem could be raised, since the protocol model relies on a synchronized internal states and keys between sensors and tags. Hence, for synchronization reasons, the tag has to run each session for a small fixed period of time. In case it does not receive the message expected to be received (a request after the authentication step, a second part of the PIN code, etc..), it switches off automatically even if the session has not ended properly. The tag returns to use the last available session key and state (in case the loss is done in the key establishment step) or to use the same session key and the next state in other steps of the protocol.

3.3.4 Security Mechanisms

The secrecy and authentication of the master key rely on the unpredictable property of the generated derived keys. Since we are using a block cipher based PRNG, the unpredictability of the derived keys is assured when the threshold N (related to the number of PRNG outputs) is not yet reached, within the same initial state. The strong notions of secrecy should be also assured by our proposed protocol, assuming the presence of a strong adversary having already captured the secret key materials Km and St. The forward secrecy supposes that the adversary is not able to guess the previous master key. The knowledge of the master key as described in our generation function ($Kd_i, St_i \leftarrow \mathcal{G}(Km, St_{i-1})$) supposes the knowledge of at least two variables (e.g., the previous St and the Kd). We will see in the next chapter that after enforcing the security of the internal state, the adversary $\mathcal{A}$ (after compromising the system) cannot obtain the previous Km, even if $\mathcal{A}$ knows the new St and predict all Kd. The backward secrecy, which supposes that the adversary is not able to guess the next master key, is also verified. However, it is assured only in a modified model of adversary, consistent with RFID environment.

In the sequel, we focus on the definition of the threshold N. Recall that this threshold exists due to the deterministic characteristic of the PRNG. Once N is reached, the outputs

may enter a cycle. This cycle is predefined in classical designs of block ciphers, e.g., Cipher Block Chaining (CBC). It is typically equal to $2^{n/2}$, where n is the length of the key. However, in more elaborated block ciphers, it is not obvious to extract this cycle. In the following section, we propose a block cipher based PRNG as a keystream generator. The cycle of this keystream generator is analyzed and defined to adapt it to the KEDGEN2 protocol.

3.4 Solitaire Keystream as a PRNG

In the KEDGEN2 protocol, we propose to use a modified version of the Solitaire keystream algorithm, as an implementation of the PRNG to generate the derived keys. The Solitaire keys could be used following the one time pad principle to encrypt messages. These keys are generated synchronously in each party, without interaction between sensors and tags. Note that the Solitaire algorithm was originally designed by Schneier [161] as a manual cryptosystem (i.e., without the need to rely on electronics or computers).

The Solitaire algorithm can be considered as part of the lightweight solutions for RFID systems (cf. Section 2.4.2). Two main reasons motivate the choice of Solitaire: the similarities between humans and low-cost devices capacities in computations and the adaptation of game cards to key generation systems. According to a recent comparative study [91], it is shown that people and tags share a similar set of memory properties. For example, RFID tags are able to store only short secrets of 32-128 bits in a volatile memory, while human beings can maintain about seven decimal digits in their immediate memory [127]. Humans and low-cost RFID systems also share a similar level of computation capacities. Nevertheless, tags are more capable than humans at performing logical operations (AND, OR, XOR), and at choosing random values. The Solitaire keystream algorithm is extremely simple in design as it was originally designed as a manual cipher. The generation of the keystream requires only the resource a deck of bridge cards, and so the algorithm is extremely lightweight. The low computation complexity of the algorithm is much envied, and is one of the main reasons we want to adopt it as a PRNG on-board of RFID low cost devices.

In the sequel, we describe the Solitaire keystream algorithm, expected to be used in our KEDGEN2 authentication protocol.

3.4.1 The Solitaire Keystream Algorithm

Solitaire generates its keystream using a deck of cards. Each card in the deck (54 cards, including the 2 jokers) corresponds to a number 1 to 54. We could for instance use the bridge order of suits: clubs (1 - 13), diamonds (14 - 26), hearts (27 - 39), spades (40 - 52), Joker A = 53 and Joker B = 54. To initialize the deck, we have to arrange the cards in the initial configuration representing the initial state. The initial state in this instance corresponds to an element of the permutation group S_{54}, meaning a state of 54 cards. It is

recommended that the initializing state of the deck be randomly chosen in order to achieve a maximum of unpredictability. To produce a single output from the keystream algorithm, the following steps should be achieved:

1. Find the A joker and move it one card down. If the joker is the bottom card of the deck, move it just below the top card.
2. Find the B joker and move it two cards down. If the joker is the bottom card of the deck, move it just below the second card. If the joker is one up from the bottom card, move it just below the top card.
3. Perform a triple cut. That is, swap the cards above the first joker with the cards below the second joker.
4. Perform a count cut. That is, look at the bottom card and count down from the top card that number. Cut after the card that you counted down to, leaving the bottom card at the bottom position.
5. Find the output card (number). To do this, look at the top card. Count that number of cards from the top. The obtained card, is the output card. If the cards is a joker, restart Step 1.

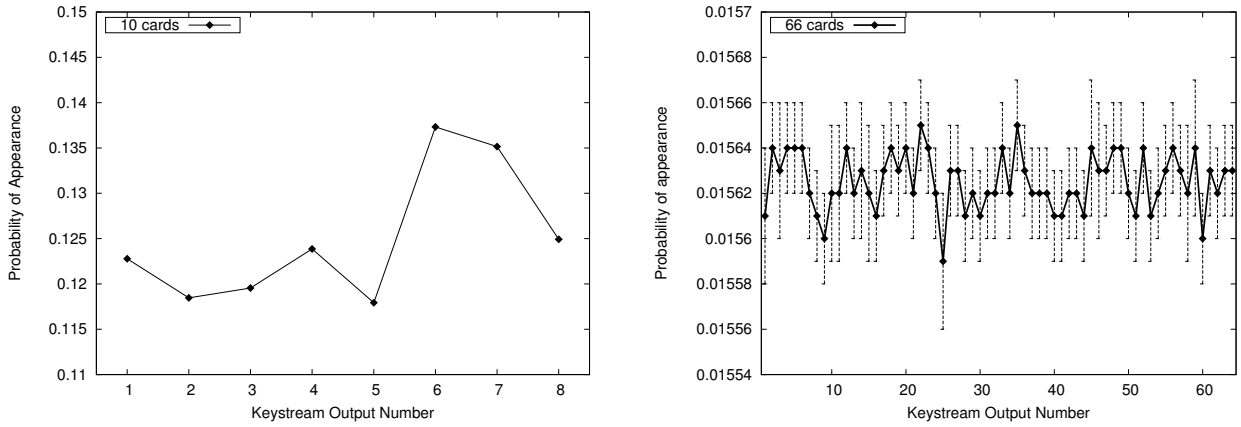
A keystream of numbers can be generated by repeating the procedure above. Solitaire designer defines an optional input named *passkey*, which is used for scheduling the order of the deck. To obtain the same order of the deck, both communication parties should share the same *passkey*. The use of this input is like performing the four first steps of Solitaire, but instead of extracting an output number (Step 5), the Step 4 is performed another time, based on the first character of the *passkey*. These operations are repeated once for each character of the *passkey*. As stated before, Solitaire keystream is intended to be used as a PRNG for the KEDGEN2 protocol. Since an automated model checking tool will be used to verify the KEDGEN2 protocol (cf. Chapter 4) and this model uses a symbolic approach that could not deal with computational properties (i.e., the cryptographic primitives of the PRNG), it is crucial for us to study the probabilistic characteristics of Solitaire (see the following section), mainly the cycle length of the keys generations in order to adapt the protocol accordingly (the threshold length N). Finally, since a PRNG goal is to ensure the unpredictability of its generated outputs, the correctness of the PRNG can be measured with statistical tests that are applied to the output keystream (cf. Section 4.4).

Uniform Distribution Property of Keystream Outputs

We start by demonstrating the uniform distribution property of the Solitaire keystream outputs. Figures 3.3 (a) and 3.3 (b) are the probability distribution plots among the possible

keystream output values using a deck of 10 cards and 66 cards (including the jokers) respectively, with 95% confidence interval.

Figures 3.3 (a) and 3.3 (b) respectively show the distribution of the values using a version of Solitaire with 10 cards and 66 cards (including the jokers). To generate these figures, we have used 1000 different sequences, composed of 100,000 keystream outputs. Each point of the graphs corresponds to the mean value of the obtained card number occurrence.



(a) 10 cards: 8 cards + 2 jokers

(b) 66 cards: 64 cards + 2 jokers

Figure 3.3 – Distribution of the Solitaire keystream numbers

We observe that the uniform distribution property emerges when the number of cards n gets larger. Indeed, as can be seen from Figure 3.3 (a), he probability distribution for a deck of 10 cards lies in the range between 0.118 and 0.137. This is to be compared with the perfect uniform distribution score $1/8 = 0.125$. For a deck of 66 cards, the discrepancy between the actual probability distribution values and the perfect theoretical value $1/64 = 0.015625$ becomes smaller (as shown in Figure 3.3 (b)). The corresponding probability distribution range for 66 cards is between 0.01559 and 0.01565. Note that the results for confidence interval are too small to be shown in the plot of Figure 3.3 (a), since they are in the order of 0.002% for the mean. These results will be verified by the NIST test in the next Chapter.

Solitaire is a block permutation based algorithm. The keystream output starts repeating itself after a cycle is reached. The cycle length of the keystream is a crucial piece of information since key recovery (e.g., using exhaustive key search) becomes possible once an adversary is able to predict the keystream outputs (cf. Appendix A, for more details).. In the remainder of this section, we model the functions of Solitaire using group theory to investigate a solution to detect its state cycle.

3.4.2 Solitaire Model in Group Theory

One natural way of modeling the Solitaire keystream algorithm is to give a group theoretic point of view. This line of investigation was extensively used in algebraic cryptanalysis of the Advance Encryption Standard (AES) [42]. In [151], the authors also used this approach in their investigation of the Solitaire algorithm.

In this theoretical framework, the state function $F(t)$ at t'th keystream number generation step is an element of the permutation group S_n, where n is the number of cards in a deck. The state function F is composed of four transformations: F_1, F_2, F_3, F_4 which correspond to the four steps of the keystream algorithm. Clearly, if a cycle of length m exists, we would have

$$S(t+m) = S(t), \quad o(t+m) = o(t)$$

from some t'th number onwards. Here, $o(t)$ denotes the t'th number in the output keystream, and $S(t)$ the corresponding Solitaire state. Thus, the task of detecting a cycle in the keystream is equivalent to finding a repetition of the Solitaire state at some regular time intervals. This approach turns out to be computationally feasible for us when the number of cards is less than 16 (i.e. $n \leq 16$, including the jokers, cf. Section 3.4.3).

Let $o(t)$ be a keystream element within a cycle, and $S(t)$ the corresponding Solitaire state. The cycle length which contains $o(t)$ is equal to the size of the orbit generated by $< S(t) >$ acted under the transformation F. In [151], the authors gave some algebraic descriptions of the orbits generated by individual actions F_i. At present, a complete algebraic description of those orbits generated by $< F = F_1, F_2, F_3, F_4 >$ does not exist in the literature. The best that is possible to do theoretically is to write down the trivial bound (i.e., the maximum cycle length is necessarily less than the group order $|S_n| = n!$).

There are some evidences from our simulations shown in the next section, which suggest that the orbit sizes are growing exponentially with respect to n.

3.4.3 Cycle Detection of Solitaire Keystream

The security of the Solitaire keystream based PRNG (to prevent key-recovery attacks) depends on the cycle length of a keystream. It is considered as secure if the best key recovery attack is computationally infeasible (requires a number of queries or a running time too large to make the attack practical). Thus, to make key recovery by exhaustive key search computationally hard, we should calculate the cycle value and prevent the adversary from reaching it. In this section, we propose a probabilistic cycle detection method.

Cycle Detection Algorithm

The principle of the cycle detection algorithm is: if a cycle of length m exists, we would have:

$$S(t+m) = S(t), \quad o(t+m) = o(t)$$

from some t'th number onwards. Here $S(t)$ denotes the state of keystream generation at t'th step, and $o(t)$ the t'th number in a keystream output. For a fixed initial configuration, we could output $S(t), o(t), t \geq 1$ in a given file.

The cycle detection in Algorithm 1 takes a keystream as an input, and outputs a cycle length when it is found. We have implemented the algorithm in Perl, as Perl is extremely efficient in tasks such as line pattern grep, and pattern mining within a keystream file. The algorithm starts by randomly selecting a line L_1 at position i_1 where $1 \leq i_1 \leq \alpha$. Here, α is an algorithm parameter that will be explained later. The algorithm then tries to pattern match among the rest of the lines, and finds $L_2, ..., L_\beta$ at the positions $i_2, i_3, ..., i_\beta$ within the keystream, which are identical to L_1, i.e.

$$S(i_1) = \cdots = S(i_\beta) \text{ and } o(i_1) = \cdots = o(i_\beta).$$

The random selection step in Algorithm 1 is justified because any keystream number is equally likely to be in a cycle, cf. Section 3.4.1. The algorithm parameter β is a stop counter, which stops the algorithm after β number of matches are found. This parameter is designed to increase the algorithm efficiency, as now the algorithm may well terminate before the end of the keystream is reached. In fact, computation efficiency turns out to be crucial in exhaustive search scenarios as those explained in the cycle detection results.

The parameter β is also significant in the following sense: the algorithm is able to terminate and concludes that a cycle has been found after β matches, because the probability that a non-cycle keystream number appears at equal distance β times is approximately $(\frac{1}{n})^\beta$, where n is the number of cards. Thus, if we choose $\beta = 8$ when $n \geq 4$, then the chance that a *fake* cycle is detected instead of an authentic cycle is negligible. The approximation $1/n$ is valid because of the uniform distribution property among all the keystream outputs, with respect to the number of cards.

The parameter α is usually chosen to be less than or equal to `keystream_length`$/\beta$, so that the algorithm would not terminate before β matches are found in the keystream file. The parameter γ has the range $1 \leq \gamma \leq \beta$. It is an in-built parameter of the algorithm because sometimes a cycle does not start right after the first state repetition. The cycle verification step (Line 8 in Algorithm 1) starts only after γ matches are found. For example, the parameter values we have used to detect the cycle for $n = 10$ cards are $\alpha = 200,000, \beta = 5, and\ \gamma = 2$.

Algorithm 1 Cycle Detection Algorithm

Input: A keystream file of which each line is composed of a state $S(t)$ and a Solitaire output $o(t)$

The algorithm parameters α, β, γ

Output: A cycle Length if the algorithm succeeds

1: Randomly select a line L_1 at the position i_1 in the keystream file, where $1 \leq i_1 \leq \alpha$
2: Sequentially search for repetitions of L_1 in the keystream file
3: **if** No Matches are Found **then**
4: Goto Step 1
5: **else**
6: Record positions $i_2, i_3, \cdots, i_\beta$ of those lines that are identical to L_1, or until the end of the keystream is reached
7: **end if**
8: **if** $i_\gamma - i_{\gamma-1} = i_{\gamma+1} - i_\gamma = \cdots = i_\beta - i_{\beta-1}$ **then**
9: **return** Cycle Length $:= i_\beta - i_{\beta-1}$
10: **else**
11: Test fails
12: **end if**

Cycle Detection Results

We present the cycle detection results based on Algorithm 1. Table 3.1 and Table 3.2 below summarize our findings. Table 3.1 contains cases where an exhaustive search is feasible. By an exhaustive search, we mean that the cycle detection in Algorithm 1 is applied to the keystream generated by every initial cards position on a deck of n cards. There are exactly $n!$ permutations which correspond to these positions. At the level of coding, we have integrated GNU GSL library [1] with the Solitaire C code [161] for the benefit of using the GSL permutation functions during the initial state/permutation rotation stage. The exhaustive search computations are performed on a PC with Ubuntu OS 32 bits and Intel Core i7 / 2.50 GHz of CPU. It is worth noting that the number of cards here, includes the two jokers for position controlling.

The exhaustive search of cycles is only feasible when the number of cards is lower than 10 ($n < 10$). The time-stamp shows that the computations in case $n = 9$ took 39 days fo each of the 9! = 362,880 possible configurations of cards. At this speed, the estimated computation time for $n = 10$ would be 195 days to calculate all the cycles of the 10! cards positions (i.e., all possible initial states), whereas for 16 cards it would take more than 3 years. Storage complexity is another issue. Recall from Algorithm 1, the execution of the cycle detection algorithm requires the storage of a keystream file (i.e., states and corresponding outputs). The length of the keystream should be at least equal to the cycle length multiplied by β (recall β is the number of matches one needs in order for the program to terminate). For example,

in the case $n = 17$, the estimated max cycle length is 76,403,613 (cf. Equation (3.1)). If one chooses $\beta = 10$, the length of the keystream is: $10 * 76,403,613$. This is equivalent to a 60 Gb textfile. Files of this size exceed the file size limit for a 32 bits machine. We tried the same tests in a 64 bits machine, where the textfiles could be larger defined. We found that for 17 cards, some lines matches are found. However, these results do not define a cycle (i.e. the distance between the matches vary). Thus, more cases need to be checked with sufficient computation time to make a conclusion about *Max.* or *Min.* periods. Here again, the problem of timing is limiting the test cases and when the number of cards gets lager, it turns to be undoable to store hundreds of Gb for each initial configuration. As a result, we are only able to run the cycle detection algorithm for the cases $4 \leq n \leq 16$.

Table 3.1 – Exhaustive cycle search

Nb. Cards n (including 2 jokers)	Nb. Config: $n!$	Min. Cycle	Max. Cycle	% Min. Cycle	% Max Cycle
4	24	5	5	100%	100%
5	120	7	14	13.3%	45.8%
6	720	8	54	4.3%	36.3%
7	5,040	4	74	<0.1%	86.8%
8	40,320	5	936	<0.1%	40.1%
9	362,880	30	2,808	<0.1%	97.9%

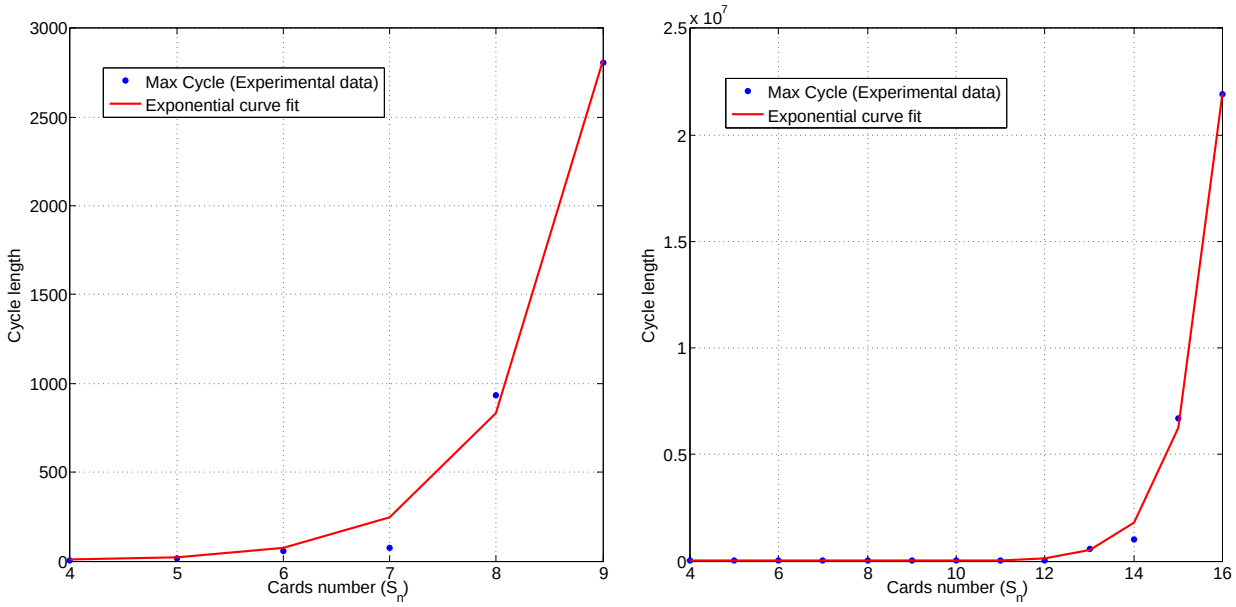
As a first observation, the Solitaire keystream may output different cycle lengths depending on the initial state of the deck (differently from classical block ciphers). We have included in Table 3.1 the percentages of the minimum and maximum cycles for all all possible positions of the state ($|S_n| = n!$), with respect to the state length n.

The maximum cycle length grows as n grows, and the growth rate turns out to be exponential (see next paragraph). The minimum cycle lengths however, do not seem to be correlated, or have a functional dependence on n, at least when n is small. The minimum cycle length for each n occurs extremely rare as shown from Table 3.1 when $n \geq 6$. On the other hand, the percentage of the maximum cycle length occurrences for each n is much higher than any other cycle length occurrence percentages. This observation also leads us to the belief that if we perform cycle detections on a set of randomly selected cards positions for the case $n \geq 10$, there is a high probability that the maximum cycle length of the set is also the global maximum cycle length.

According to this, for the cases $n \geq 10$, the cycle detection algorithm is performed on a set of randomly picked states (cards positions), i.e., a fraction of all cards positions in a same state length. Tables 3.2 contains the random configuration search results for the cases $n = 10, ..., 16$. It shows the percentage of the *Max. Cycles*. The *Min. Cycle* length for each configuration, in this case, represents a minimum fraction within the allotted number of

Table 3.2 – Random cycle search

Nb. Cards n (including 2 jokers)	Nb. Config Used	Max. Cycle	% Max Cycle
10	1,054,596	10,221	75.9%
11	246,663	7,543	39.9%
12	195,943	33,058	87.2%
13	119,373	554,526	91.2%
14	1841	1,013,519	-
15	1343	6,702,967	-
16	17	21,936,204	-



(a) Exhaustive search

(b) Probabilistic search

Figure 3.4 – Evolution of the maximum cycle length

searches. Thus, we do not take them into account as a global minimum cycles, since they could be lower when an exhaustive searches is handled. Also, we did not record the percentage of maximum cycle length occurrences for the cases $n = 14, 15, 16$ on the ground that the number of samples (i.e., cards positions) checked is too small to produce any significant statistics.

Extrapolation Results

Extrapolation results are based on *Max. length* cycles. Figures 3.4 (a) and 3.4 (b) respectively show the evolution of the maximum cycle with the exhaustive cycles search and

the random cycles search. The analytical fit is mostly superposed on the experimental data in each figure. Results show that the evolution of the period grows exponentially with respect to n. According to Figure 3.4 (a), the coefficients a and b expressed with 95% confidence bounds, are shown in Equation (3.1):

$$f(n) = a \exp^{b*n}, where\ a = 0.02925\ ;\ b = 1.276 \tag{3.1}$$

As a conclusion, the estimated period length can be used to rekey the initial state S_n, every $f(n)$ Solitaire generation. The aim is to avoid, as long as possible, reaching the orbit size of a given initial state.

3.4.4 The Solitaire Keystream Generator

Motivated by the above results, we adapt the Solitaire keystream to be used into our KEDGEN2 protocol as follows.

As shown in Section 3.3.1, the modeled generator $\mathcal{G}en$ consists of two algorithms. The first algorithm $\mathcal{S}$ is probabilistic and is responsible for generating the master key. This algorithm is performed the first time the initial state is set. The master key could be the optional *passkey* of Solitaire to shuffle the current state or directly a new calculated initial state St_0. The first solution requires more computations as Solitaire algorithm has to be run l times, where l is the length of numbers composing the *passkey*. The second solution requires the sending of the state in several messages (if it is a long state length). For the sequel, we choose the second case. The second algorithm of $\mathcal{G}en$, in this case, is the iterative deterministic generation algorithm representing the four permutation functions $< F = F_1, F_2, F_3, F_4 >$ of Solitaire. The Solitaire algorithm computes in each iteration, from an initial key St_{i-1} and optionally a *passkey*, an output Kd and a new state St_i. For instance, for $i \geqslant 1$, the

Algorithm 2 Adapted Solitaire

Input: St_0, St_{i-1}, St_i, Max_cycle
Parameter: t: number of keystream generation
Output: Refresh State entry St_0 after Max_cycle

```
Initialization: St_{i-1} ← St_0, t ← 0
while Session not ended do
    while t < Max_cycle do
        St_i ← Solitaire (St_{i-1})
        St_{i-1} ← St_i
        t ← t + 1
    end while
    St_{i-1} ← St_0 , t ← 0
end while
```

Solitaire algorithm takes as input the St_{i-1} (optionally including the *passkey* perturbator) to generate Kd_i and St_i as: $Kd_i, St_i \leftarrow \text{Solitaire}(St_{i-1}, passkey)$. Note that when the *passkey* is not used, it is set to void. We associate with Solitaire a re-keyed encryption scheme, which establishes a new key, representing the new initial state St_0 to be used in every new session or whenever the cycle limit or threshold (estimated in Section 3.4.3) is reached in the same session. More formally, the adaptation of Solitaire within the KEDGEN2 protocol can be described in Algorithm 2. For sake of simplicity, Algorithm 2 only considers the progress of the states.

Recall, that our aim is to minimize the advantage (i.e., the likelihood) of the adversary $\mathcal{A}$ to compromize the security of Solitaire keys using the data $\mathcal{A}$ recovered in each transaction. The refreshment of the states when the threshold is reached is a birthday attack [21] defense. In the next chapter, we will evaluate our proposals: the KEDGEN2 protocol and the adapted Solitaire PRNG.

Conclusion

EPC Gen2 suffers from a flawed protocol when RFID readers want to access the tag memory. In addition, the related pseudo-random number generator suffers from weak security assumptions to guarantee the predictability of the outputs. In this chapter, we presented a new key establishment and derivation protocol for Gen2 systems, which are in use in different environments, such as healthcare. We have shown that our proposed protocol, namely KEDGEN2, focuses on the privileged access to the tag memory, where keys and other sensitive information could be stored (e.g., related to the bearer of the tag). In addition, since passive RFID tags have similar capabilities as humans [91], and the card shuffling of Solitaire keystream algorithm has low complexity, we proposed to use and adapt this keystream as a PRNG for KEDGEN2 protocol. We demonstrated that the modified (or adapted) version of Solitaire can be easily integrated in the KEDGEN2 protocol to generate derived keys. In fact, it is possible to integrate the initial state (i.e., initial cards position) as a master key, to update the state every keystream generation and to update the master key every N output round, representing the cycle length. We estimated the maximum cycle length of Solitaire to determine an approximative value of the threshold N. We explained that we have not used the minimum cycle length because these cycles do not seem to be correlated, or have a functional dependence on n, at least when n is small. Our extrapolation results show that the maximum cycle length grows exponentially with respect to the state (or deck) length. Our two contributions will be evaluated in the next chapter with two different approaches. A symbolic approach to check the KEDGEN2 protocol and a statistical approach to verify the randomness criteria of the modified Solitaire generator outputs.

CHAPTER 4

Verifying the Security of the KEDGEN2 Protocol

"Crypto algorithms are like locks. Even if you have the perfect lock, things can go wrong." – Cas Cremer

Introduction

KEDGEN2 is a new protocol for passive RFID devices that has been proposed in Chapter 3. The KEDGEN2 protocol handles particularly the flawed security model of EPC Gen2, when readers execute privileged commands such as accessing the tag memory. Additionally, an adapted version of Solitaire cipher is proposed as a pseudo-random number generator embedded in the protocol to ensure long term secure communication. We have shown that its particularity as a hand cipher could meet the constrained environments of RFID passive devices. Nevertheless, even when using a strong cryptographic algorithm as part of a communication protocol, this does not by itself guarantee that the system will be secure. Indeed, good cryptography can be used in a bad way due to bad designed protocol. Thus, a major need of proved and validated security RFID protocols appears. These proofs may be done following a computational or a formal approach aided by some tools developed for this purpose. Unfortunately, this was not systematic in most work dealing with RFID security where the security goals were demonstrated *intuitively*. In fact, recent cases of authentication techniques were reported flawed few time after their publication, cf. Chapter 2. For that aim, we propose in this chapter the use of a formal approach, named also symbolic approach, to verify the KEDGEN2 protocol with respect to strong notions of secrecy and authentication. We prove formally that after being amended, the protocol is secure under these properties. As symbolic approaches do not verify, cryptographic primitives, an empirical approach is used to check for the randomness criteria of the Solitaire PRNG. The remainder of this chapter is organized as follows. We first present the Constraint-Logic based Attack Searcher (CL-AtSe) tool to check for the requested security properties, its specification language and its output format. Then, we provide the specification of the KEDGEN2 protocol and the evaluation results of our protocol under the CL-AtSe tool. Next, a comparative discussion with relevant works related to our proposal is provided in Section 4.3. Finally, we show the statistical tests of the proposed Solitaire pseudo-random number generator.

4.1 Verifying the Security of KEDGEN2

We use model checking techniques to verify whether a security property holds in a finite state machine. The goal is to find errors (e.g., logical flaws) leading to attacks against the protocol implementation in accordance with the security assumptions provided in Section 3.1. Automated reasoning is highly desirable to avoid errors associated with hand-written proofs. If a security design of complex systems is verified successfully by an automated tool, it increases the confidence of the system users.

In the sequel, the verification tool we use is presented. Then, some notions about the HLPSL input language and the structure of the outputs are given.

4.1.1 Checking Tool

There is a number of successful protocol verification tools that are supporting algebraic reasoning, e.g., the extended ProVerif [101], Maude-NRL Protocol Analyzer (MaudeNPA) [66], On the Fly Model Checker (OFMC) [19] and Constraint-Logic based Attack Searcher (CL-AtSe) [178]. They use different models and proof techniques. For example, extended ProVerif is based on tree automata and Horn clauses techniques. MaudeNPA is based on rewriting techniques and backward search of *bad* states. OFMC is based on a state space exploration and CL-AtSe is based on a constraint solving technique. All of these tools use the *Attack construction* technique. Each tool has some strengths and weaknesses [48]. As a candidate, we use the CL-AtSe protocol analyzer, the probably most mature tool using the constraint solving technique [99]. The tool was part of the AVISPA project [13] that has been extended recently by the Avantssar [12] project. In this paper we use the new version of CL-AtSe [15] to verify our protocol. The AVISPA platform is a suite of applications commonly used for formal specification and automated validation and verification of cryptographic protocols. It is composed of several modules: A translator called HLPSL2IF [37] is used to transform a given HLPSL specification to a low level specification IF and four different verification tools (CL-AtSe, OFMC, SAT based Model-Checker (SAT-MC) [14] and Tree Automata based Protocol Analyzer (TA4SP) [184]) to analyze the IF specifications. Among the available AVISPA tools, we have chosen CL-AtSe for its protocols analysis taking in account the algebraic properties of XOR operator. In addition, Cl-AtSe only limits the number of sessions without bounding the number of messages sent by the adversary as well as their size. Thus, the verification of protocols remains decidable. This is not the case in other model checking tools such as FDR [107], Murϕ [128] and Maude [53]. These latter limit arbitrarily both the message size and the number of sessions of the protocol, limiting the state space of exploration.

The CL-AtSe tool. Constraint-Logic based Attack Searcher consists in running the protocol or set of services in all possible ways by representing families of traces with positive or negative constraints on the intruder knowledge, on variable values, on sets, etc. Thus, each run of a

(a)
```
role A(param) played by ag
def= local L
init Init
transition
evt.0 ^ evt.1 = | > act.1
evt.2 = | > act.2
...
evt.n = | > act.n
end role
```

(b)
```
role S (param)
def= const C
composition
R1 (param-1) ^ . . . ^ Rn
(param-n)
end role
```

(c)
```
role environment (param)
def= const C
intruder_knowledge = IN
composition
S1 (param-1) ^ . . . ^ Sn
(param-n)
end role
```

(d)
```
goal
secrecy_of sec_kn
end goal
```

Figure 4.1 – HLPSL main elements. (a) Basic role structure. (b) Session role structure. (c) Environment role structure. (d) Secrecy in the goal section.

service step consists in (1) adding new constraints on the current intruder and environment state, (2) reducing these constraints down to a normalised form for which satisfiability is easily decidable, and (3) deciding whether some security property has been violated up to this point. CL-AtSe does not limit the service in any way except for bounding the maximal number of times a service can be iterated, in the case such an iteration (or loop) is specified by the user. Otherwise, the analysis might be non-terminating on secure services and only heuristics, approximations, or restrictions on the input language could lift this limitation. In our protocol verification, we specify three consecutive iterations (i.e., sessions). This number is indeed representative of different steps of our protocol and is sufficient to check the properties we wish to verify.

4.1.2 HLPSL Format

The protocol and the security properties are specified in the High Level Protocol Specification Language (HLPSL) [36]. HLPSL is a specification language for formalising protocols and security goals based on Lamport's Temporal Logic of Actions (TLA) [102]). The language, developed in the context of the AVISPA project [13], is a role-based language. Roles can be `basic` (e.g., agent roles) describing the action of a legitimate participant during the execution of the protocol or `composed` (e.g., session and environment roles) describing scenarios of basic roles to model an entire protocol run. Finally, the HLPSL language allows to specify the knowledge and capacities of the adversary model.

Basic roles. Figure 4.1(a) shows how the basic role is generally structured. Each basic role declares its name (`A`), its initial information or parameters (`param`) and the agent playing the role (`ag`). The basic role can declare a set of local variables (`L`). The `init` section assigns the initial values to the local variables, if required.

The `transition` section describes changes of the agent state. It consists of a trigger (e.g., `evt.2`) and an action (e.g., `act.2`) to be performed when the trigger event occurs. The =|> symbol separates the two phases.

Composed roles. Composed roles combine basic roles, either in parallel or in sequence. HLPSL defines two composed roles: the session role and the environment role. Actions, in composed roles, are not defined in a `transition` section like in basic roles. Rather, a `composition` section is defined to instantiate other roles `Ri` or `Si`, with sets of parameters `param-i`, that run in parallel (cf. Figures 4.1(b) and 4.1(c)). The session role, referred by `S` in Figure 4.1(b), instantiates in its composition section the basic roles and the different channels relating them while the environment role instantiates in its composition section all the sessions to be run (`Si`). The environment role is called the main role, as it declares the global constants (`C`) and defines the intruder knowledge denoted by `IN`.

Security properties. HLPSL provides an independent section to declare the security properties required, named `goal`. The goal declaration can be done either by using predefined macros of the predefined security properties (secrecy, weak authentication, strong authentication) or by using Linear Temporal Logic formulas [102]. We are interested in the predefined secrecy and strong authentication properties. We use the predefined secrecy property to check whether the secrecy of the key is maintained in a given session and to check (with a slight change of the specification) whether the forward/backward secrecy defined in Sections 3.2.3 and 3.2.4 are guaranteed in last/next session respectively, after compromising the system in a given session. We use also the authentication property to validate the goals defined in Section 3.2.1.

- Secrecy is modeled by means of the goal predicate *secret(Km,sec_km,Sensor,Tag)* standing for the value of term Km is a secret shared only between agents *Sensor* and *Tag*. The secrecy property is violated every time the adversary learns a value that is considered as secret and that he is not allowed to know (i.e., Km).
- Authentication is modeled by means of the goal predicates $witness(A, B, id, T1)$, $request(B, A, id, T1)$ and $wrequest(B, A, id, T1)$. These predicates are used to check if an instance of a role is right in believing that its peer is present in the current session. It is done by agreeing on a certain value (e.g., $T1$) which is typically fresh. The predicates always appear in pair and have the same third parameter. This third parameter id is the identifier of the authentication goal and it is used in the goal section of the HLPSL code. There exists two definitions of authentication: weak and strong authentication.
  1. $witness(A, B, id, T1)$ for a strong or weak authentication properties of A by B on $T1$, declares that agent A is witness for information $T1$. This goal will be identified by the constant id in the goal section;
  2. $request(B, A, id, T1)$ for a strong authentication property of A by B on $T1$, declares that agent B requests a check of the value $T1$. This goal will be identified by the constant id in the goal section;

3. $wrequest(B, A, id, T1)$ similar to request, but for a weak authentication property. It is used to specify an authentication goal with no replay protection.

Strong authentication is an extension of the weak authentication which precludes replay attacks. We can thus conclude that, if strong authentication is achieved, then $T1$ has not been previously received by B in a given session.

```
role sensor(...)
…
/\ witness (A,Tag',sensor_tag_kd1,
   keygen(KM',succ(KM',InState')))
…
end role

role tag(...)
…
/\request
(Tag,Sensor,sensor_tag_kd1,keygen
(KM.InState))

end role
```

```
role environment(...)
…
tag_sensor_kd1 :protocol_id
…
end role

goal

  authentication_on
sensor_tag_kd1

end goal
```

Figure 4.2 – Strong authentication property definition

Each property is added to the honest role and to the goal section. It is identified by `protocol_id` type. Figure 4.2 shows a declaration of a strong authentication property of the *sensor* by the *tag* on the value of $Kd_1 = keygen(KM', succ(KM', InState'))$ declaring that agent *sensor* is witness for the value of Kd_1 and that agent *tag* requests a check of the value Kd_1. This goal is identified by the constant $sensor\_tag\_kd1$ in the `goal` section.

4.1.3 Output Format

After the verification process, the output describes the results, and under what conditions they have been obtained (e.g., Figure 4.5 shows the verification results of the KEDGEN2 protocol). The output format is nearly common to all tools of the framework. In the `SUMMARY` section, it indicates if the protocol is safe, unsafe, or if the analysis is inconclusive. In a second section titled `DETAILS`, the output shows conditions under what the protocol is declared safe/unsafe/inconclusive. If a security property of the input specification is violated then the tools output a warning, some details about the analysis (e.g., whether the considered model is typed or untyped), the property that was violated (e.g., authentication), statistics on the number of explored states, and, finally, an `ATTACK TRACE` that gives a detailed account of the attack scenario. If no attack was found, then similar information is provided without announcing any violation and attack trace.

4.1.4 Our HLPSL Specification

The specification of both the protocol and the security goals is described into four HLPSL sections: the sensor, the tag, the environment roles and the goal. Figure 4.3 shows the

specification with mutual authentication and secrecy of the master key goals. The generation function $\mathcal{G}$ is specified by two functions *keygen* and *succ*. The first function generates the derived keys and the second one generates accordingly the new state (i.e., $InState$). Figure 4.4 shows the specification of the protocol to handle the forward secrecy. Note that for the cases of forward and backward secrecy, we have slightly changed the specification (compared to Figure 4.3) as the AVISPA tool only supports a single execution trace. Thus, we have modelled the execution of two consecutive iterations in order to show whether leaking a secret during session i helps the adversary to obtain secrets from session *i-1* for forward secrecy or session *i+1* for backward secrecy. In the following, we detail the specification and evaluation results.

4.2 Evaluation Results of the KEDGEN2

For each security property defined in Section 3.2 and specified in Section 4.1.4, we show in this section the results obtained after the evaluation of our protocol specifications under the CL-AtSe tool.

4.2.1 Mutual Authentication

Figure 4.5a shows the results of the evaluation of the mutual authentication property. To obtain these results, we specify an iteration of the protocol with legitimate roles and give to the adversary the knowledge of the generation functions, roles and standard commands used in the KEDGEN2 protocol communication (cf. Figure 4.3). In the HLPSL language, the authentication property is specified using the *witness/request* predicates. These predicates are used to check if an instance of a role is right in believing that its peer is present in the current session. We use the HLPSL strong authentication definition to require that a given value is accepted by the sensor in exactly the same session in which it was proposed by the tag. We add these predicates to the tag and sensor transactions to evaluate the authentication of each of the two roles and prevent man-in-the-middle and replay attacks. The tool finds no attack violation of the strong authentication property. This strong property guarantees the resilience to man-in-the-middle and replay attacks in which the adversary could impersonate one of the two parties.

4.2.2 Secrecy of the Master Key

Figure 4.5a shows the results of the secrecy property evaluation. We recall that the secrecy of the master key when shared securely between the tag and the sensor is mathematically maintained since the security threshold N of distinguishability is not reached. In other words, the adversary is not able to detect correlations between the outputs of $\mathcal{G}$, named the derived

```
role sensor(A: agent,
  DataBase: (agent.text.message.text) set,
  Snd,Rcv : channel (dy)) played_by A def=

  local
    Tag:agent, InState:message,
    Tkn,KM,NewKM: text, State:nat

  init
    State := 0

  transition

    0.State=0 /\ Rcv(start) =|>
      Snd(A.reqID) /\ State':= 1

    1.State=1 /\ Rcv(Tag')
    /\ in(Tag'.Tkn'.InState'.KM',DataBase)
    =|>
    State':=0 /\ NewKM':= new()
    /\ DataBase':=cons(Tag'.Tkn'.
     succ(KM',InState').KM',
     delete(Tag'.Tkn'.
     InState'.KM',DataBase))
    /\ Snd(xor(NewKM'.Tkn',
     keygen(KM',succ(KM',InState'))))
    /\ State':=2
    /\ witness (A,Tag',sensor_tag_kd1,
    keygen(KM',succ(KM',InState')))
    /\ secret(KM',sec_km1,{A,Tag'})

    2.State=2
    /\ Rcv(keygen(KM',InState'))
    /\ in(Tag'.Tkn'.InState'.KM',DataBase)
    =|>
    request(A,Tag,tag_sensor_kd1r,
    keygen(KM'.InState))

end role
```

```
role tag(Tag,Sensor: agent,
  InState : message,
  KM : text,
  Tkn:text,
  Snd,Rcv: channel(dy)) played_by Tag def=

  local
     State : nat

  init
     State := 0

  transition

     0.State=0 /\ Rcv(Sensor.reqID) =|>
     State':=1 /\ Snd(Tag)
     /\ InState':= succ(KM,InState)

     1.State=1 /\ Rcv(xor((KM'.Tkn),
     keygen(KM.InState))) =|>
     Snd(Tag.keygen(KM'.succ(KM'.InState)))
     /\InState':=succ(KM'.succ(KM'.InState))
     /\ State':=0
     /\request(Tag,Sensor,sensor_tag_kd1,
       keygen(KM.InState))
     /\witness(Tag,Sensor,tag_sensor_kd1r,
       keygen(KM'.succ(KM'.InState)))

end role
```

```
role environment() def=

  const
    sensor,tag1,tag2: agent,
    token1,token2: text,
    instate1,instate2: message,
    km1,km2:text,
    reqID: text,
    succ,keygen: function,
    r2t,t2r: channel (dy),
     a:agent,
    sec_km1,sensor_tag_kd1,
    tag_sensor_kd1r : protocol_id


  intruder_knowledge={reqID,succ,
   keygen,sensor,tag1,tag2,keygen}

  composition
    reader(sensor,
       {
          tag1.token1.instate1.km1,
          tag2.token2.instate2.km2
       },
       r2t,t2r)
     /\ tag(tag1,sensor,instate1,
        km1,token1,t2r,r2t)
     /\ tag(tag2,sensor,instate2,
        km2,token2,t2r,r2t)

end role
```

```
goal
    secrecy_of sec_km1
    authentication_on sensor_tag_kd1
    authentication_on tag_sensor_kd1r
end goal


environment()
```

Figure 4.3 – HLPSL specification of KEDGEN2 protocol to check for strong authentication and secrecy.

```
role sensor(A: agent,
  DataBase: (agent.text.message.text) set,
  Snd,Rcv : channel (dy)) played_by A def=

   local
     Iter:nat,Tag:agent,
     InState:message,Tkn,KM,NewKM: text,
     State:nat

   init
    State := 0

   transition

    0.State = 0 /\ Rcv(start) =|>
    Snd(A.reqID) /\ State':= 1

    1.State = 1 /\ Rcv(Tag')
    /\ in(Tag'.Tkn'.InState'.KM',DataBase)
    =|>
    State':=0 /\ NewKM':= new()
    /\ Snd(Tag'.NewKM'.Tkn'.
    succ(NewKM',succ(KM',InState')).
    xor(NewKM'.Tkn',
    keygen(KM',succ(KM',InState'))))
    /\ DataBase' := cons(Tag'.Tkn'.
    succ(NewKM',succ(KM',
    succ(KM',InState'))).NewKM',
    delete(Tag'.Tkn'.InState'.KM',
    DataBase))

    /\ secret(KM',sec_km1,{A,Tag'})


end role
```

```
role tag(Tag: agent,
  InState : message, % InState = instate0
  KM : text,Tkn:text,
  Snd,Rcv: channel(dy)) played_by Tag def=

   local
     Reader: agent,
     State : nat

   init
     State := 0

   transition

     0.State=0 /\ Rcv(Reader'.reqID) =|>
     State':=1 /\ Snd(Tag)
     /\ InState' := succ(KM,InState)

     1.State=1 /\ Rcv(xor((KM'.Tkn),
     keygen(KM,InState))) =|>
     State':=0
     /\ Snd(Tag.keygen(KM',
     succ(KM',InState)))
     /\ InState' :=
     succ(KM',succ(KM,InState))


end role
```

```
      role environment() def=

           const
             sensor,tag1,tag2: agent,
             token1,token2: text,
             instate1,instate2: message,
             km1,km2:text,
             reqID: text,
             succ,keygen: function,
             r2t,t2r: channel (dy),
             sec_km1, sec_resp,
             sensor_tag_kd0,
             tag_sensor_kd1 : protocol_id


      intruder_knowledge={reqID,succ,keygen
      ,sensor,tag1,tag2,keygen}

           composition
           reader(sensor,
             {
                 tag1.token1.instate1.km1
                 ,tag2.token2.instate2.km2
             },
             r2t,t2r)
           /\ tag(tag1,instate1,km1,
           token1,t2r,r2t)
           /\ tag(tag2,instate2,
              km2,token2,t2r,r2t)
      end role
```

```
goal
    secrecy_of sec_km1
end goal


environment()
```

Figure 4.4 – HLPSL modified specification of KEDGEN2 protocol to check for forward secrecy.

```
INPUT V7-1-ProtocolAuthentifSecrecy.if
SUMMARY NO_ATTACK_FOUND
DETAILS TYPED_MODEL
BACKEND CL-ATSE VERSION
2.5-8_(February_23th_2011)
STATISTICS TIME 44 ms
TESTED 105 transitions
REACHED 34 states
READING 0.04 seconds
ANALYSE 0.00 seconds
```

(a) Authentication and secrecy evaluation

```
INPUT V7-6-forward-orig-chiff.if
SUMMARY ATTACK_FOUND
GOAL:secrecy_of_sec_km1(km1,set_53)
DETAILS TYPED_MODEL
BACKEND CL-ATSE VERSION 2.5-8
_(February_23th_2011)
STATISTICS TIME 28 ms
TESTED 10 transitions
REACHED 6 states
READING 0.01 seconds
ANALYSE 0.02 seconds
```

(b) Forward secrecy evaluation (original specification)

```
INPUT V8-forward-chiff.if
SUMMARY NO_ATTACK_FOUND
GOAL:secrecy_of_sec_km1(km1,set_53)
DETAILS TYPED_MODEL
BACKEND CL-ATSE VERSION 2.5-8
_(February_23th_2011)
STATISTICS TIME 24 ms
TESTED 27 transitions
REACHED 17 states
READING 0.01 seconds
ANALYSE 0.01 seconds
```

(c) Forward secrecy evaluation (modified specification)

```
INPUT V7-6-backward-chiff.if
SUMMARY ATTACK_FOUND
GOAL: secrecy_of_sec_km1(n3(NewKM),
set_55)
DETAILS TYPED_MODEL
BACKEND CL-ATSE VERSION
2.5-8_(February_23th_2011)
STATISTICS TIME 928 ms
TESTED 16 transitions
REACHED 12 states
READING 0.05 seconds
ANALYSE 0.88 seconds
```

(d) Backward secrecy evaluation

Figure 4.5 – Evaluation results.

keys. The model checker is used in our evaluation to confirm that the adversary is not able to desynchronize the two participants and replay some messages to reconstruct the master key (and with that, the secret messages encrypted using such a key). To verify the secrecy of the master key, we specify with HLPSL a single instance of the protocol with legitimate roles and give the adversary the knowledge of inner working of the system (cf., Figure 4.3). Secrecy is modeled by the mean of the goal predicate $secret(Km, sec\_km, Sensor, Tag)$ standing for the value of term Km is a secret shared only between agents Sensor and Tag. The secrecy property is violated every time the adversary learns the value Km that is considered as secret.

4.2.3 Forward Secrecy

Figures 4.5b and 4.5c show the forward secrecy evaluation results. To prove forward secrecy, we consider a setting in which the tag and the sensor try to establish a new master key $NewKm$ using the previous master key Km. Once $NewKm$ has been established, we reveal to the adversary the internal states $NewKm$, $InState$, and Tkn of both the tag and the sensor. Our goal is to prove that this knowledge is not sufficient to enable the adversary to compute the previous Km. We prove first that the original specification of our protocol (cf. Figure 4.3) does not provide forward secrecy. This is shown with the results in Figure 4.5b. The analysis of the attack trace shows that after establishing and sending the new master key to the tag (i.e., $M = (NewKm||Tkn) \oplus Kd_1$ where $Kd_1 = \mathcal{G}(Km, InState_1)$), the adversary obtains Km in the next generation of $InState$ ($InState_2 = \mathcal{G}(NewKm, InState_1)$) relying on the knowledge of $NewKm$ and Kd_1. The countermeasure is to hide the generation of $InState_2$ by values which are not deduced by the adversary. This way, the adversary cannot obtain the key Km. In fact, by changing $\mathcal{G}(NewKm, InState_1)$ to $\mathcal{G}(NewKm, \mathcal{G}(Km, InState_1))$, we use a double generation of the initial state depending on values that cannot be computed by the adversary (i.e., Km). This modification is shown in Figure 4.4. The evaluation results in Figure 4.5c show that the modified version satisfies the forward secrecy property even under the hypothesis of a complete compromise in the following sessions.

4.2.4 Backward Secrecy

Figure 4.5d shows the results of the backward secrecy property evaluation. We consider two executions of the sensor. One execution in which the tag and the sensor establish a master key Km and another one where we reveal to the adversary the last secrets (i.e., $Km, State, Tkn$) of both the tag and the sensor. The goal is to verify if this knowledge is sufficient to enable the adversary to compute the new master key $NewKm$ related to this execution. The results show that the protocol is insecure. CL-AtSe finds an attack on the secrecy of the new master key. Indeed, if the adversary follows all the messages sent in the network, it is possible to reconstruct the following master key $NewKm$ because the new derived key used to encrypt the message of refreshment (i.e., 3^{rd} pass in the Figure 3.2) can be computed. The new derived keys are based on the previous secrets that the adversary has already gained, and once obtaining these secrets, the adversary takes all the power of the target tag itself. He can trace it at least during the authentication immediately following the attack. However, assuming a strong adversary is unrealistic in RFID environments, since in a typical RFID system, tags and readers operate only at a short communication range and for a short period of time. This attack can be avoided if the adversary only misses one successful key establishment transaction after it has compromised the secrets. This scenario is likely to happen, as tags are in motion. The property related to such scenario is known to *restricted* backward security through key insulation [165, 105]. As a result, assuming a weaker but realistic adversary model that captures the capabilities of a real world adversary,

if this latter does not eavesdrop on the tag continuously after the time of corruption, then it will be not possible to predict the next refreshed derived keys. This assumption is compliant with works in [105, 17, 55, 156], stating that perfect tag security and privacy under a strong adversary cannot be achieved using the limited resources on RFID tags.

4.3 Comparative Discussion

In this section, we present the most relevant works related to our proposal as well as a comparative discussion between them.

Several papers e.g., [179, 32, 78, 31, 95, 16] have analyzed forward security and other communication faults in RFID systems at various levels of formality. Some of them define the authentication and secrecy (named also privacy) in computational model, typically in terms of games. For example, in [179], the authors define two security protocols to assure authentication and forward secrecy using the universal composability framework. After detecting a synchronisation problem, a new series of protocols was proposed by the same authors. The last version in [32] improves the protocol and includes a verification of the backward secrecy property using the same framework. The authors in [78], use a game-based approach to prove the robustness of an RFID protocol against a man-in-the-middle adversary. They do not propose a new protocol to be applied on constrained tags but a new method to prove the security of the OSK protocol [98] that they combine with a mechanism to synchronize the internal state of the tag and reader. The resulted protocol can be applied on RFID tags supporting hash functions. The security of the protocol is proved using the computational model of CryptoVerif verification tool combined with some handwritten proofs to overcome the limitations of the tool regarding desynchronization and forward privacy verification.

Works in [31], [95] and [16] have used the symbolic model to formally verify the security properties. These works are closest to ours. The advantage of using the symbolic model, as our work does, is its ease to automatically prove the security of cryptographic protocols and to clarify these complex protocols with provided definitions of formal languages. In view of this, the authors in [31] use the applied pi calculus language with the ProVerif automated verification tool and apply their proposed techniques to the OSK protocol [98] in order to formally prove the untraceability and forward privacy properties. The proposed technique, which consists in the concept of frame independence between sessions, meet our security goals. However, the proposed verification technique is applied only on one class of protocols that the authors call *single step* identification. Furthermore, this new technique is applied on protocols with two distinct hash functions. This is only possible on tags computationally strong enough to use such functions. These two criteria make the solution different from our proposal, i.e., our proposal uses more steps for both identification and authentication in the context of Gen2 tags (without hashing capabilities). In [95], the authors use the automated verification tool FDR (Failure Divergence Refinement). The work can be compared to ours as it also uses a

model checking tool to verify the secrecy and authentication of an RFID protocol. However, the use of a hash based scheme added to a pseudo-random number generator to implement the protocol presents a different solution model. Moreover, as opposed to our proposal, authors do not consider in their work strong secrecy notions such as forward and backward secrecy that handle linkability between the sessions, Finally, as seen in Section refsub:formalVerif, the FDR tool limits the state space exploration, since it bounds messages and sessions. This way, possible attacks may appear in an unexplored part of the state space. In [16], the authors propose a new protocol that assures mutual authentication and privacy which they define as anonymity and forward untraceability. The verified property of forward untraceability is different from forward secrecy. Authors define the protocol as attacked when the attacker detects twice the same hash result, which means detecting the same tag. Whereas in our case, an attack is shown when an attacker obtains the secret keys of last sessions of communicated keys for a given tag. Furthermore, the authors propose a one-way hash function as a solution for a secured RFID protocol and use the AVISPA OFMC tool for automated verification.

Differently from all cited protocols achieving a formal verification of their proposals in the symbolic model, our proposed KEDGEN2 protocol is applied on highly constrained tags such as Gen2 since it is only based on a pseudo-random number generators. Recall that in our work, we assume that the use of hash functions is beyond the capabilities of low-cost RFID tags, as reported in [112, 26] for EPC Gen2 tags. KEDGEN2 achieves mutual authentication, and forward and backward secrecy in different conditions. We prove these properties in the AVISPA framework using the Constraint-Logic based Attack Searcher (CL-AtSe) automated verification tool. This tool is based on an *attack construction* methodology attempting to find vulnerabilities using algebraic properties of protocols.

It is worth noting that in [32] and [179], the authors apply their protocols on the same category of Gen2 tags. However, they use the universal composability framework, which is based on the computational model.

To summarize, Table 4.1 shows the different aspects that differentiate our KEDGEN2 protocol to such previous efforts.

Table 4.1 – Comparison between recent related work using various models of formality

Main characteristics	[179]	[32]	[78]	[31]	[95]	[16]	KEDGEN2
Formal model	Computational	Computational	Computational	Symbolic	Symbolic	Symbolic	Symbolic
Framework	Universal composability	Universal composability	CryptoVerif	ProVerif	FDR	AVISPA (OFMC)	AVISPA (CL-AtSe)
Forward secrecy	√	√	√	√	–[1]	–	√
Backward secrecy	–	√[2]	–	–	–	–	√[3]
Authentication	√	√	√	√	√	√	√
Cryptographic primitives	Pseudo-random generator. e.g., shrinking generator	PRNG[4]	Three distinct hash functions	Two distinct hash functions	Hash function + PRNG	Hash function	PRNG
Application on highly constrained tags	Possible	Possible	Not possible	Not possible	Not possible	Not possible	Possible

√: Checked by the authors – not checked by the authors

[1] A different definition of forward secrecy is checked named forward untraceability

[2] Checked under the same adversary used for verifying the forward secrecy

[3] Checked under a weaker adversary used for verifying the forward secrecy

[4] PRNG: Pseudo-random Number Generator

4.4 Evaluation Results of the Solitaire PRNG

We have seen in Chapter 3 that the uniform distribution property emerges when the number of Solitaire cards n gets larger. This makes possible to distinguish its output from a truly random generator, in cases of small number of cards. We study in more detail the outputs randomness of Solitaire. To verify whether the Solitaire generator produces numbers appearing to be random, we use the NIST statistical test suite for cryptographic applications [154]. In the following section, we detail our statistical tests.

4.4.1 Statistical Testing Tool

First, we present the statistical testing tool used in our analysis. Our results are generated from the National Institute of Standards and Technology (NIST) statistical test suite for random and pseudo-random number generators for cryptographic applications [154].

The statistic that is generated from each of the NIST runs is based on a P-Value approach. It consists in computing a test statistic for a sequence s and its corresponding probability value (P-value). The P-value is the probability of obtaining a test statistic as large as the one observed with a random sequence. The sequence s passes a statistical test if the P-value $\geq \alpha$, and fails otherwise. α is the significance level. Typically, α is a value in the interval [0.001, 0.01]. For example, if we get conventionally P-values < 0.01, this means that a sequence is unlikely to be random. If we get a P-Value equal to 0.95, this means that 95% of the sequences produced by a true RNG would look less random than our sequence.

Note, that in addition to P-Value approach, other testing approaches exist in the literature. We name the *Threshold Values* and the *Fixed Ranges* approaches. The first one shows some false positive and negative regarding the evaluation of the sequences [168]. The second one imposes the use of a fixed range of values and a pre-computed significance level α. The P-Values have the added advantage that they do not require the significance level specification. Once a P-value has been computed, this latter can be compared to an arbitrary α.

Presentation of the results

The NIST test suite produces a table containing a summary report for the file of bits it tests (i.e., in our case, the file contains the Solitaire outputs rendered into digits). Each test in the NIST suite runs over a set of bits from this file. The table consists of:

- Ten columns labeled C1 through C10: each of these columns represents the number of tests that has a P-value in the corresponding range (i.e., the range from 0 to 1 is divided into ten equal-length segments called *bins*). A perfect RNG would have P-values uniformly spread over the range 0 to 1.

- P-VALUE column is a P-value of P-values. The documentation that comes with the suite indicates that: "If P-Value [the number in the column labeled P-VALUE] >= 0.001, then the sequences can be considered to be uniformly distributed".
- PROPORTION column indicates the number of P-values that are above the 0.01 confidence interval. Thus, it is acceptable for a few individual tests to fail. The test suite will indicate a problem by flagging the PROPORTION number with an "*".
- Name of the test column that is related to that row. NIST presents a number of evaluation tests. Here, we evaluate Solitaire outputs with the three following tests:
  - Frequency (monobit) is to determine whether the number of ones and zeros in a sequence are approximately the same. This would be expected for a truly random sequence.
  - Block Frequency is to determine whether the frequency of m-bit blocks in a sequence appears as often as would be expected for a truly random sequence.
  - Cumulative Sums is to determine whether the maximum of the cumulative sums in a sequence is too large or too small. In other words, it is used to detect if there are too many zeroes or ones at the beginning of the sequence.

  These tests are the most significant for our case study. For example, the Frequency (or monobit) test is the basic test. It should success otherwise, there is no need to run the subsequent tests. The Block frequency is a pertinent test because we are dealing with blocks of bits as derived keys to hide the messages sent using the KEDGEN2 protocol.

Data Management

A large amount of data is generated for the purpose of the NIST evaluation of the Solitaire outputs. We conducted various tests for the original Solitaire algorithm, varying the deck size n. The input files contain usually contain billions of bits. Hence, files and data management for this huge amount of data were of a real concern. We have adhered to the following test procedures: (1) fix the deck size, so that the representation of the card could be completely rendered in bits (e.g., in the 16 cards version of Solitaire, each card is represented in 4 bits); (2) generate necessary amount of keystream output data in decimal; (3) transform the decimal data into bits based on the size of the deck, and align all the outputs into one line of bits; (4) input the bit file into the NIST testing suite (version *sts-2.1*).

4.4.2 Statistical Tests

A test on Solitaire 34 cards

Figures 4.6 and 4.7 show NIST test summaries over a set of data produced by the Solitaire with 34 cards (32 cards + 2 jokers). An input file of 1 billion bits is used. During the test,

100 individual sequences are constructed, each of which consisting of 10 million bits. For each sequence, three basic and essential tests described in the previous section are performed (Frequency, Block frequency, and Cumulative sum). The two block values considered in the block frequency tests are16-bit and 800-bit.

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST
15	9	11	8	6	10	14	12	5	10	0.419021	0.9900	Frequency
80	11	5	0	4	0	0	0	0	0	0.000000 *	0.5100 *	BlockFrequency
14	9	10	4	11	15	8	11	10	8	0.455937	0.9900	CumulativeSums
14	12	9	15	10	5	13	5	10	7	0.249284	0.9900	CumulativeSums

Figure 4.6 – NIST test outputs for 34 cards (16-bit block size)

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST
15	9	11	8	6	10	14	12	5	10	0.419021	0.9900	Frequency
24	12	10	15	8	8	6	6	8	3	0.000216	0.9800	BlockFrequency
14	9	10	4	11	15	8	11	10	8	0.455937	0.9900	CumulativeSums
14	12	9	15	10	5	13	5	10	7	0.249284	0.9900	CumulativeSums

Figure 4.7 – NIST test outputs for 34 cards (800-bit block size)

For this test, NIST indicates that the minimum pass rate for each statistical test is approximately equal to 0.96 for a sample size equal to 100 binary sequences. As shown in Figure 4.6, both the frequency test and the Cumulative Sums test have passed. However, the block frequency test fails when the block size is 16 bits. After re-sizing the block up to 800 bits (cf. Figure 4.7), the Block frequency test is successful.

A general test

We have also performed the NIST tests for Solitaire with 10 cards, 18 cards, 34 cards and 66 cards, all considered with the two jokers. These values correspond respectively to the 3, 4, 5, and 6 bits output values. As before, an input file of 1 billion bits is used for each deck size. We have used the same test configuration as the previous test case of 34 cards. Table 4.2 contains the test results. We have highlighted successful test entries in bold. Recall that the minimum pass rate for each statistical test is approximately equal to the proportion of 0.96 for a run of 100 binary sequences.

Test Results for the case of $n = 10$ are not statistically good. Here is an example of Solitaire output when $n = 10$:

```
1 3 5 5 5 8 3 3 5 5 5 3 1
```

Cards | Tests	Frequency	Block Frequency16 bits	Block Frequency800 bits	Cumulative sums
10	0.0	0.0	0.0	0.0
18	**0.98**	0.0	0.35	**0.98**
34	**0.99**	0.51	**0.98**	**0.99**
66	**1.0**	0.13	**0.88**	**1.0**

Table 4.2 – Proportion of passed tests on different cards number (including the jokers)

The frequent succession of the same pattern `3 5 5 5` indicates the reason of failure for the frequency test and block frequency test (16 bits block).

From a deck of size $n = 18$, the sequences pass the frequency test and the cumulative sum test. However, only the version of 32 cards passes the block frequency test when the block size is equal to 800 bits. This problem of block frequency can be explained in the following sample output ($n = 18$):

```
6 16 16 14 14 ..... 12 15 1 15 13 9 7 7 3 5 10 6 16 16 16
```

In the beginning and the end of this sequence, we see a succession of `6 16 16`. Converting this subsequence to a block of 16 bits, we have `0110111111111****`, this block pattern frequently occurs in the rest of the keystream file, which explains the failure of block frequency test, although the block repetition for $n = 18$ is much less frequent than the case $n = 10$.

We may conclude here that when the block size is larger, the probability of a block repetition is lower. Hence, this produces a higher chance of passing the block frequency test.

4.4.3 Difference between original and adapted Solitaire version

The adapted Solitaire algorithm is a slight variation of the original Solitaire algorithm described in Chapter 3, Section 3.4. The original keystream algorithm has a fixed initial state, whereas the adapted Solitaire requires a random or a pseudo-random change of initial state once a maximum cycle length as predicted by formula (3.1, in Section 3.4.3) is reached (i.e. start the Solitaire with a random initial state after a Max. cycle is reached). In order to show the difference between the original and adapted Solitaire algorithms, we conduct two different tests: a keystream generations with Solitaire of 10 cards and Solitaire of 18 cards (including the jokers).

Comparison using a deck of 10 cards

Figure 4.8 is a plot comparing the original Solitaire against the adapted version. To generate this plot, 1000 experiments are run over a population of 100,000 Solitaire outputs. The adapted Solitaire algorithm produces a better uniform distribution result than the original

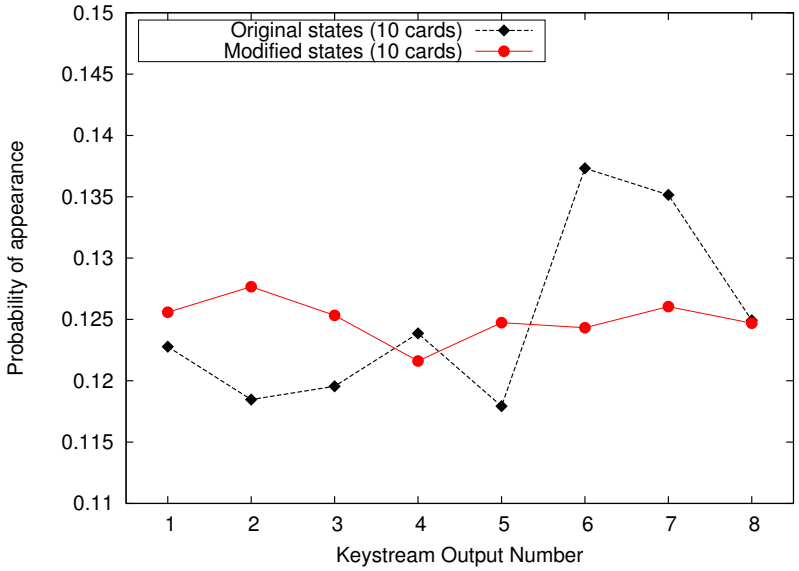


Figure 4.8 – Distribution of the Solitaire keystream (original and modified states)

version. The probability distribution range for the original Solitaire lies between 0.118 and 0.137, whereas the adapted Solitaire gives a probability distribution range between 0.122 and 0.128.

The NIST statistics confirm these resuts. They also show an enhancement from the adapted Solitaire keystream algorithm, see Table 4.3. Again here, the numbers highlighted in bold indicate the success of a particular randomness test, with a minimum pass rate approximately equal to the proportion of 0.96 for a run of 100 sequences.

Solitaire (10 cards) | Tests	Frequency	Block Frequency	Cumulative sums
Original states	0.0	0.0	0.0
Updated states	0.44	**0.97**	0.44

Table 4.3 – Proportions of passed tests in the original and modified Solitaire - 10 cards version

Comparison using a deck of 18 cards

Using formula (3.1) from Section 3.4.3, a max. cycle for $n = 18$ is estimated to occur after approximately 270 million keystream outputs [1]. For the NIST test, an input file consisting of 800 million bits is used. 800 individual sequences are constructed from this file, each consisting of 1 million bits. For the block frequency test, we have set the block size equal to 1000 bits.

[1] We consider a state of 18 cards due to the practical length to perform the state update

Solitaire (18 cards)	Frequency	Block Frequency	Cumulative sums
Original states	**0.98**	0.93	**0.98**
Updated states	**0.99**	0.93	**0.99**

Table 4.4 – Proportions of passed tests in the original and modified Solitaire - 18 cards version

Table ?? summarises the test results. In particular, we have that the P-values are more less the same for both original and modified Solitaire. The proportion values for Frequency and Cumulative sums tests are slightly enhanced in the adapted version. Nevertheless, the Block frequency test remains unsuccessful for both versions.

From the experimental results obtained in this chapter, we may conclude that the Solitaire keystream (original and adapted) algorithm does not operate as a typical block cipher (cf. Section 3.3.1), which has a fixed keystream cycle $2^{n/2}$ (n is the length of the state). In this case, the adversary can easily calculate the cyclic repetition of states and predict the next outputs (i.e., birthday attack [21]). The Solitaire keystream cycles vary in length, depending on the initial state of cards positions, making the cycle search more difficult for the adversary. We have also shown that updating the Solitaire initial state after a maximum cycle length enhances the uniformity of the Solitaire outputs (cf. Figure. 4.8). However, there is not enough convincing evidences at present for us to conclude that the adapted Solitaire with the maximum cycle length is a better candidate than the original Solitaire to be used as a PRNG in RFID systems. In fact, the optimum period for rekeying the Solitaire deck (i.e.,the initial state) remains to be found, in order to show better statistics.

Conclusion

With the goal of addressing security concerns on the use of the Gen2 RFID technology in sensitive domains, we presented the specification and verification of a key establishment and derivation protocol for Gen2 systems. The KEDGEN2 protocol achieves secure data exchange between tags and readers, based on a key generation model adapted to Gen2 RFID tags. The generated keys are used in the proposed protocol as one time encryption keys. To guarantee the security of the protocol, the generation function has to respond to a number of properties, including the resilience against key recovery and the indistinguishability of the derived keys. We described the steps of our protocol and verified the expected properties under the presence of an active adversary. The current version of the protocol guarantees the properties of mutual authentication and forward secrecy. Backward secrecy is also verified under a weaker but realistic adversary model, consistent with typical RFID environments. As an implementation of the modeled PRNG in the KEDGEN2 protocol, we proposed to adapt the Solitaire keystream algorithm. We analyzed the output characteristics of Solitaire keystream and found that the uniform distribution property emerges when the number of

Solitaire cards n, gets larger. We also observed that the modified version, when updating the initial state every maximum cycle generation, clearly enhances the uniformity of the generated keystream numbers and slightly improves the test statistics of the outputs. However, the exact period of rekeying the Solitaire deck (i.e., the initial state) remains to be found, in order to show better statistics. These results turn out to assure long term impredictable keys.

Part II

Privacy-enhanced RFID Middleware in the EPCglobal Networks

CHAPTER 5

State of the Art

"We lose our individuality, because everything we do is observable and recordable." – Bruce Schneier

Introduction

We have seen in the first part of this dissertation that there are some security and privacy threats in the low level interface linking readers to tags and related to the front-end of the EPCglobal network. In this second part, we will focus on the back-end of the same RFID infrastructure. We concentrate on the middleware component, which is the central point for both data collection and request configuration. In fact, the system at this point is likely to suffer from parameter manipulation and eavesdropping leading to privacy concerns. Since the data collected from RFID tags to the back-end applications contain sensitive information (e.g., in healthcare domains), data protection against privacy violations is a need that must be addressed in such open and collaborative context. The confidentiality of these data should be guaranteed from both external users breaking the system and malicious insiders (e.g., curious applications). The following sections of this chapter outline the main preliminaries and related work needed to present our contribution. In a first part, we begin by presenting the architecture of the EPCglobal network and its basic components. Then, we present the Filtering & Collection middleware and its interface with a focus on its limitations regarding privacy control. A survey on relevant work related to security and privacy in the middleware is given. In a second part, we detail the privacy dimensions appearing in known regulations and guidelines. Then, we outline the relevant privacy models, with a particular focus on PrivOrBAC, which is based on the Organization-Based Access Control model (OrBAC). This model handles most of the privacy dimensions appearing in the international regulations, that will be also presented.

5.1 EPCglobal Network Architecture

EPCglobal network is currently one of the predominant standardization efforts of the RFID community. It is represented by a set of global technical standards providing real time data

about individual items and linking them to the global Internet. In the following section, we present the main components of this RFID architecture.

5.1.1 EPCglobal Network Components

Figure 5.1 shows the relationship between several EPCglobal standards [11], from a data flow perspective. In this Figure, we do not consider individual identification and reading of tags (cf. Part I). Each processing step, played by hardware or software components of a typical IT architecture, is mediated by an interface governed by an EPCglobal standard. From the bottom of the figure to the top, we can identify the following elements:

- RFID Readers make multiple observations of tags, when these tags are in the read zone. The observations are performed via the air interface protocol typically the UHF (Ultra High Frequency) Class1, Gen2 in the case of passive EPC tags. Readers may perform data dissemination, tag ID filtering, time aggregation (e.g., entry and exit events) and space aggregation (e.g., multiple readers antennas can be logically grouped into a single source at the reader level). Readers also support writing to tags and external triggers.
- Reader Interface defines the control and delivery of raw tag read from Readers to the middleware to communicate them to the local business information systems. Events at this interface say: "Reader A saw EPC X at time T".
- Filtering & Collection Middleware (F&C) decouples readers and applications and provides additional aggregation and filtering functionalities. It brings the concept of logical readers (i.e., addition of different readers) making data intelligible for information system. Once the relevant tag data are produced, the F&C middleware combines these data in a report, so that the EPCIS and local information systems can work with these data.
- Application Level Event Interface defines the control and delivery of filtered and collected tag data to the capturing application role. It represents a single interface to the

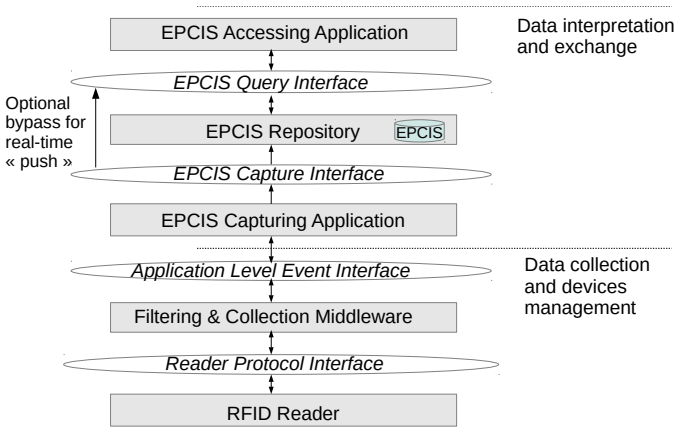


Figure 5.1 – EPCglobal network components: Roles and Interfaces

potentially large number of readers. Events at this interface say: "At Location L, between time T1 and T2, the following EPCs were observed", where the list of EPCs has no duplicates and has been filtered by criteria defined by the EPCIS capturing application.

- EPCIS Capturing Application is a F&C middleware client. It subscribes to the middleware from where it receives uninterpreted RFID events. It is in charge of translating these uninterpreted events into meaningful business events, e.g., checking for exceptional conditions and present information to a human operator. This role may be complex by involving the association of multiple F&C events with one or more business events.
- EPCIS Repository provides persistence for EPCIS events. It represents a convenient standardized mechanism to store and access relevant business events and to exchange information with trading partners. It is directly related to the two following roles:
  - EPCIS Capture Interface, through which new events generated capturing applications are stored in the repository as EPCIS-level events. It makes them available for later query by EPCIS accessing applications. Events at this interface say: "At location X, at time T, the following contained objects were verified as being aggregated to the following containing object".
  - EPCIS Query Interface, through which historical events can be retrieved.
- EPCIS Accessing application achieves all the enterprise business processes, such as warehouse management, historical throughput analysis, aided by EPC-related data.

Finally, other roles are related to this architecture such as Discovery Services & Subscriber Authentication [11]. They are a suite of services that allow users in a wide area to find data associated with a specific EPC and to request access to the data.

The EPCglobal network shown in Figure 5.1 captures the architecture of the system at a local site or organization. Data collected in an organization from local EPCglobal network may be decentralized, supporting distributed information management. The EPCglobal network achieves this goal by linking local networks together through Internet. The resulting wide area EPCglobal network achieves a global flow of data and enables the exchange of information between organizations at the level of individually identifiable objects.

Data representation: The EPC standard allows several representations. The different EPC forms specified in the EPC tag data standard [75] are intended for use at different levels within the EPCglobal architecture framework. For example, a binary encoding is used for events in RFID tag and reader level, whereas the Pure Identity EPC URI (read only) or EPC Tag URI (read/write) are used to represent the middleware events. The Universal Resource Identifier (URI) is used for data exchange. More details can be found in Appendix B.

5.1.2 Intra and Inter Organizational Data Aggregation

Systems that manage, control and store the collected data should be secure both from external users breaking the system and from malicious insiders. In fact, external organizations may be interested in the data operated by other companies, whereas, in a same organization, different services may be interested in aggregating data related to one EPC tag or many EPC tags of one person. Thus, there are internal and external need for data aggregation, which may reveal rich information.

Reader level is generally run to manage and collect data for internal network requests [39], while EPC Information Services (EPCIS) level is used either for internal and external users queries. Regarding the middleware level, it is mainly used for internal exchange. For example, to collect all the data of a patient items in one organization, we can configure the middleware interface to aggregate the results of two different readers to obtain the patient related data in different places. Nevertheless, with the emergence of the Service Oriented Architecture (SOA) concepts, which supports intrinsic interoperability, the idea of middleware-to-middleware communication across-organizations is possible [3]. In the sequel, we focus on the middleware and its ALE interface.

5.2 The Filtering & Collection Middleware and its Interface

The RFID middleware is one of the main components of an RFID architecture (cf. Section 5.1). It should automatically or semi-automatically control, configure, monitor, and organize the devices from multiple RFID reader vendors, in order to effectively allow them to communicate over the network with the enterprise applications. Thus, the middleware design must be fully compliant with international standards, and support simultaneous communication of multiple applications with multiple RFID hardwares. RFID middleware products have widely adapted EPCglobal standards. These globally accepted standards ensure global applicability, in which EPCglobal Inc. shows a strong initiative towards overcoming incompatibilities between companies of RFID-related IT infrastructures.

5.2.1 The EPCglobal Application Level Event Interface

The F&C (Filtering & Collection) middleware of the EPCglobal network uses a single interface to a large number of distributed readers and a large number of capturing applications that may be interested in the collected data (cf. Figure 5.2). This interface is called the ALE [62] (Application Level Interface). ALE provides means for host applications (or clients) to specify, in a high-level, declarative way, what tag data they are interested in. It also provides a standardized format to report collected and filtered tag data independently from the origin of tag data or the manner it was processed with. Finally, it abstracts the sources of tag data

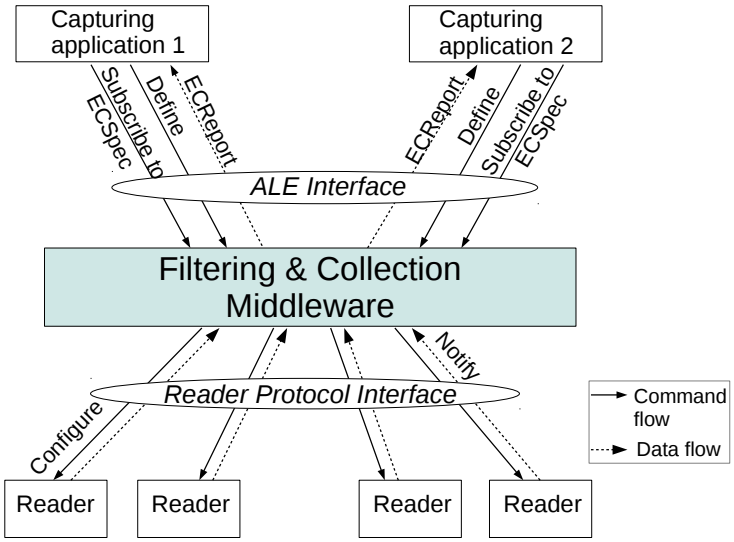


Figure 5.2 – The EPCglobal middleware

(e.g., Readers, Barcode scanners) into higher-level notions of "location". The ALE specification defines a SOAP message transport binding for the subscription communication channel and an XML and TCP/HTTP message transport binding for the notification channel.

ALE 1.1 comprises five standards APIs: (i) the Reading API, responsible for reading tags and giving reports in variety of ways, (ii) the Writing API, responsible for initializing, writing, locking and killing the tag, (iii) the tag Memory API, which has the role to define symbolic names for memory fields to be used by Reading and Writing APIs, (iv) the Logical Reader API, which has the role to define symbolic names for reader/device resources to be used by Reading & Writing APIs, and (v) the Access Control API, responsible for controlling access by clients to other API features. The two first APIs are primarily used by applications and are known as the data plane. The last three APIs are used for setup and administration and are known as the control plane. We are interested in the Reading API whose responsibility is to read tags and give reports in a variety of ways.

It is undisputed that the act of reading out one or more RFID tags constitutes a data collection. This means that existing privacy laws (cf. Section 5.3.2) also apply to the communication involving the RFID middleware responsible for configuring RFID readers for the data collection. In the sequel, we give more details about the Reading API of the middleware interface. Then, we present the limits of the ALE regarding privacy control.

Primary data types

The primary data types associated with the Reading API are the Event Cycle Specification (ECSpec) and the Event Cycle Report (ECSpec). ECSpec is the type used to define how an Event Cycle is to be calculated (i.e., specification of readers, notification latency, aggregates, filters). Event Cycle is the smallest unit of interaction between the ALE and the capturing

application. It can consider multiple physical readers as a logical reader since it deals with them from the application view. As a consequence, a standardized ECReport is generated and provided to the capturing application, cf. Figure 5.2. The ECReport contains one or more reports which are generated from one execution of an ECSpec. ECSpec and ECReport are both defined in XML format. In Section 6.2.3 of the next chapter, we describe, in more details, the components of the ECSpec, to identify the fields that deserve to be treated by privacy rules.

Finally, it is worth noting that the interaction between the client and the ALE for both tag identification and memory access are similar. For tag identification, the ALE client provides a specification ECSpec. For tag memory access, the client transmits either a specification ECSpec for reading the memory content or a CCSpec (Command Cycle Spec) for writing on the tag memory. In the two cases, the middleware executes the specification, captures the RFID data or carries out the memory access operations on the tags and returns the requested reports.

Request modes

One or more clients (i.e., capturing applications) can request the ALE interface [62] for filtered and aggregated data, via a set of methods. Each request causes the ALE engine to take an action and return synchronously a result. The ALE interface also enables clients to subscribe to events that are delivered asynchronously, by using a Uniform Resource Identifier (URI). This URI describes the client address to which the information is delivered. e.g., *subscribe(ECSpec, URI)* is the method to send a subscription request for an already defined *ECSpec*. Figure 5.3 depicts three sequence diagrams to demonstrate the use of asynchronous (*subscription/poll*) methods and synchronous (*immediate*) method.

- Asynchronous ECSpec request is performed when a client defines the ECSpec using the *define* method and subsequently, one or more clients subscribe to that ECSpec using the *subscribe* method. The ECSpec generates event cycles as long as there is at least one subscriber. In this scenario, the receiver of the ECReports can be different from the subscriber (i.e., the ALE Client). A *poll* method is like subscribing then unsubscribing immediately after one event cycle is generated. But here, the ECReports are directly returned as results of the *poll* method instead of being sent to a URI.
- Synchronous ECSpec request is an immediate execution of the ECSpec, which can be submitted using the *immediate* method. This is equivalent to defining an ECSpec, performing a single *poll* operation, and then undefining it.

In the following chapter, we will use the asynchronous mode of request to implement our privacy module.

5.2.2 Privacy Control Limitations in EPCglobal Specifications

EPCglobal has proposed an ALE API of access control to use the functionalities of the F&C middleware. This API allows administrative clients to define the access rights of other clients when using the methods and resources proposed by other ALE APIs (e.g., reading, writing to the tag memory) in the F&C middleware. The model is role based access control. A role maps to one or more permissions to a particular feature of the ALE API. A permission describes an access to a particular feature of the ALE API. Two kinds of permissions exist: the *function permissions* and the *data permissions*. The first one grants the right to use a particular method of the ALE API (e.g., define, subscribe, unsubscribe), whereas the latter grants the right to use a particular resource or data (e.g., the right to govern a particular reader or to use another ALE API).

The security mechanism of the access control API limits the accesses to critical methods (i.e., subscription to capture tag data). Nevertheless, it does not cope with the details of data aggregation and the use of filtered and combined reports within a request specification. We believe that a fine-grained security on these collected data for privacy concerns has to be handled to prevent applications, even in the same organization, to collect data that undermine people's private lives or reputations. This is the aim of our next contribution detailed in the next two chapters.

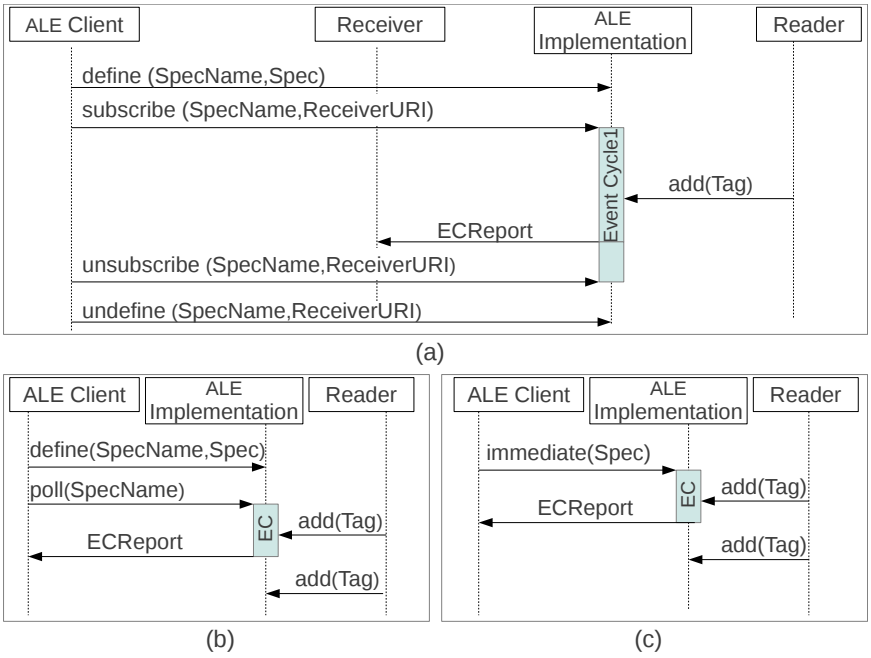


Figure 5.3 – Request methods: (a , b) for asynchronous request (c) for synchronous request

5.2.3 Related Work on Middleware Implementations

In the RFID architecture, three levels of data deserve to be considered with security and privacy aspects. We note the reader and tag events (cf. first part of this manuscript), the middleware events and the EPCIS data (after events have been captured and stored). We consider that the reader interface is part of the middleware interface as the two roles are in the "middle" of the architecture and relatively play the same role with different levels of abstraction [62] (i.e., Generally, the reader protocol is the interface in which the client can enter low level settings which are abstracted by the ALE specification).

While security research in the context of RFID has mainly focused on reader to tag communication [88, 41, 175] and RFID-based data repositories [7, 57, 159], the middleware level has not received much attention so far. Most of the middleware implementations available today are commercial-based (cf. [85] for a survey on the middleware taxonomies). There are also some middlewares which have been developed for research purposes, such as [69] and [84]. However, most widely deployed middleware products do not fully consider the security requirements when processing sensitive data. For example, several implementations from software vendors [83, 138] and specialized companies [152] and open source initiatives [69] offering RFID middleware functions, manage the access control to historical events stored into final databases. However, they do not provide treatments to enhance privacy policies in accordance with international regulations, expected to be done at the middleware level [135]. The same issue is observed in some researches for securing the middleware level. For instances, authors in [167] provide an Application Level Events Service Security Component (ALE-SSC) to strengthen security and trust in the ALE-based service middleware. The proposed mechanism is only to protect transported data between the middleware and its clients. In [166], the authors provide techniques to support context-aware access control service in the middleware. The context information, in this case, is meant to be the reliability of the requester and the environment.

Regarding the EPCglobal standard, most existing implementations support its use. In case they include security mechanisms, they explicitly provide it in the EPCIS (EPC Information Service) level [83, 138, 152] relying mostly on role based access control and encryption models. Some of these aforementioned products include security enhancements into their middleware implementations [83, 138]. However, few works focus on privacy problems. An exception is the work [84], where the authors propose to support the EPCglobal middleware via a tool called *Privacy Framework Tool* plug-in. This tool includes privacy friendly practices and audits to be applied to the proposed middleware. To the best of our knowledge, there have been neither public communication related to the used privacy policy nor information about the plug-in implementation. We also note the work in [9] that provides security and privacy features included in the middleware, that is referred as the data processing layer. In this work, the authors enforce the middleware privacy by policy based management. According to the authors, the privacy policy specifies whether an application has the right to access

RFID tag data, track it over time, and use it to generate events. It is not mentioned that the privacy policy has considered the privacy dimensions of accuracy and consent, issued from known regulations (cf. Section 5.3.2).

As a conclusion, today RFID interfaces that govern the middleware propose a role based access control as a security approach. However, they fail to address consumer privacy concerns by appropriately supporting the fair information practices principles [67]. This leads us to focus on this issue, to show the possibility of modifying the existing EPCglobal middleware standard in order to satisfy the basic principles of privacy. In the rest of this chapter, we briefly restate the basic principles of privacy and their role in today's privacy guidelines. Then, we give some relevant privacy models in the access control literature.

5.3 Privacy Issues and Regulations

A frequent definition of privacy is *the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others* [183]. The definition of not sharing or hiding information is fundamental and has been adopted for a long time. However, due to the highly dynamic nature of user data, the privacy of people focuses today on when, how, where, and specifically to whom they disclose their information [111]. This way, the issues of privacy are completely treated in different situations, where the owner of the data chooses to share information with different roles.

5.3.1 Privacy Approaches

Typically, there are two approaches to introduce privacy features: user-centric and system-centric approaches. In the user-centric approach, the user chooses the rules and settings for the behavior of information contained in the system. In the system-centric approach, rule sets and policies ordain the privacy settings [142].

Traditional solutions for granting data privacy have been aligned on anonymizing user information or preventing personal data from disclosure, e.g., by encryption means [50]. In addition, a central system in the organization controls what each individual should be allowed to collect as information. Recent work have looked towards more user-centric models that attempt to put individuals in charge of their personal information disclosure [111]. This approach adds more flexibility to manage user preferences. In addition, dealing with encrypted data (as it is traditionally done), makes query processing expensive [68], either in a database or a middleware level. Furthermore, it is not always needed to encrypt all the data since not all the collected information is sensitive, rather, the aggregation of some information may need to be protected. For example, if the list of illnesses or the list of patients in a hospital could be made publicly available, the association of specific illnesses to individual patients presents a sensitive information. Thus, there is no need to encrypt

both illnesses and patients if there are alternative ways to protect the association between them [24] or to filter them in early levels. This is what we aim to do in our work.

Since assisted healthcare systems are primarily user-based systems, the privacy settings, in this case, primarily depend on the awareness and preferences of the patient. On the other hand, the patient might not be aware of the standards and practices involved in the disclosure of information. Thus, a trade-off between the user-based and the system-centric approaches for privacy preserving could be implemented. This could develop a dynamic approach including the flexibility of the user-centric privacy settings, along with the policies and rule sets of system-centric privacy [142]. In this dissertation, we apply this approach to resolve privacy issues when collecting aggregated information. In the sequel, we name as data owner the user who specifies a set of privacy preferences. The preferences depend on the context and privacy dimensions supported by the application.

5.3.2 Regulations and Guidelines Worldwide

Choosing a privacy-protective system requires a solid grounding in foundation privacy principles. The Fair Information Practices (FIP), published by the Organization of Economic Cooperation and Development (OECD) in 1980 [67], are the most widely accepted set of guidelines for consumer privacy. These principles apply broadly to the collection and use of personal data.

The FIP principles form the basis for many of today's privacy laws, such as (i) the EU Directive 95/46/EC [135], which provides the framework for the national privacy laws of all EU-member states, and (ii) the Recommendation on the implementation of privacy and data protection principles [44], which are recommended for applications using RFID systems. Based on these principles, we define the main privacy dimensions in RFID environments, as:

1. Collection limitation: Collectors should only collect necessary and appropriate information. This should be done by lawful means and according to the data owner preferences with his knowledge or consent.

2. Purpose specification: the purpose for which data is collected and disclosed should be announced at the time of collection, otherwise should not be disclosed without the consent of the data owner.

3. Explicit consent: obtaining the unambiguous consent of the data owner before any unexpected collection of data is required.

4. Notification of the data owner (named also openness): the existance of systems containing personal data should be publicly known. This guarantees the right of the data owner to be informed when a new request for data is to be achieved.

5. Data owner access (named also individual participation): the data owner should have the right to view or query the data keeper about his personal data, and, if possible, to correct, erase or amend them.

6. Security mechanisms: reasonable security mechanisms should be designed to protect personal data from risks of loss, unauthorized access or disclosure (e.g., encryption, anonymity).

7. Collection relevance (named also Data quality): collected data should be up to date and retained with respect to the period of relevance.

5.4 The Privacy Model

There are several models of privacy in the access control literature. These models are used to represent and integrate privacy dimensions into a security policy [8, 114, 134, 185]. The security policy is generally specified according to an access control model to simplify the upgrade of existing information systems. Models like P-RBAC [134], Purpose-BAC [185] and Pu-RBAC [114] define new languages to express access contexts, additionally, they focus on the purpose entity and some other privacy requirements [67]. Here, we focus on the dimensions supported by the PrivOrBAC model [8], which extends the Organization-Based Access Control model [92] (OrBAC). PrivOrBAC reuses most of the OrBAC implemented mechanisms. For example, it manages the security policy by specifying the contexts as complex conditions to define fine grained privacy control requirements. The choice of PrivOrBAC relies on its capacity to handle most of the privacy dimensions appearing in the guidelines and recommendations previously mentioned (e.g., [67], [44]), but of course it can be substituted with another privacy model without disrupting the whole process.

5.4.1 The OrBAC Model

OrBAC is a generic and expressive access control model that extends the Role Based Access Control [158] (RBAC) model by supporting environment and dynamic parameters through the contexts. OrBAC provides a set of concepts to express the security policy and enables making distinction between an abstract policy specifying organizational requirements and its concrete implementation in a given information system. Abstract organization privileges, such as *permission*, are expressed through the predicate *Permission(org, r, a, v, c)*. It means that the organization *org* grants a permission to role r to realize the activity a on the view v in the context c. When declaring a context, a subject obtains some specific permissions and possibly some obligations or prohibitions. For instance, a subject empowered in the role *nurse* may be permitted to declare, as a context, that she is performing an *inspection*. By doing so, this subject will get the permission to have an access to the tag data that are in relation with her service. Thus, contexts are used to specify conditions.

Five kinds of contexts have been defined [49]: (1) Temporal contexts that depend on the time at which the subject is requesting for an access to the system, (2) Spatial contexts that depend on the subject location, (3) User-declared contexts that depend on the subject purpose, (4) Prerequisite contexts that depend on characteristics that join the subject, the action and the object, and (5) Provisional contexts that depend on previous actions the subject has performed in the system.

These elementary contexts can be combined to define composed contexts by using conjunction, disjunction and negation operators: &, $\oplus$ and $\neg$, e.g., if c1 and c2 are two contexts, then c1 & c2 is a conjunctive context defined by the following rule: $Hold(org, s, \alpha, o, c1\&c2) \leftarrow Hold(org, s, \alpha, o, c1) \wedge Hold(org, s, \alpha, o, c2)$.

5.4.2 Privacy Contextual Management

PrivOrBAC models the explicit consent as a context, the purpose as a user declared context and the different view levels as an accuracy of the objects. For example, for dimensions of purpose and accuracy, the data owner decides why its data should be disclosed to a defined recipient (purpose) and under which view the data are disclosed (accuracy).

The Consent dimension

We define the consent preference view and the consent context, as follows.

- The consent preferences are stored by users in a consent preference view (*cp*). Each object in this view has four attributes: *dataowner*, *Recipient* (who receives the data related to the object), *Target* (the requested object), and *NeedConsent* (a Boolean parameter whose value is true when the consent is needed).
- The consent context takes into account the data owner preferences and/or optionally notifies him when his personal information is accessed. The data owners can define their policies by specifying that context preference should be checked before granting the access. Two cases are identified. The first case is when the consent is needed (*NeedConsent = true*). In this case, the data owner response is modeled by a built-in predicate *Consent_response*. That is, if *org* is an organization, *s* is a subject, *do* is a data owner, *resp* $\in$ *{accept, deny}*, then *Consent_response(org, do, s, cp, resp)* is the response returned by the data owner to the organization. The second case is when the consent of the data owner is not required before revealing his private data to the recipient. In this case, the *NeedConsent* attribute is set to *false*. The access decision can be made without waiting for the *Consent_response*.

  The user consent context is specified as follows:
  $Rule_{consent} : \forall org \in Org, \forall s \in S, \forall \alpha \in A, \forall o \in O, \forall cp \in$

$O, Hold(org, s, \alpha, o, Consent\_context) \leftarrow Use(org, cp, Consent\_preference) \wedge Recipient(cp, s) \wedge Target(cp, o) \wedge Data\_owner(cp, do) \wedge (\neg NeedConsent(cp) \vee Consent\_response(Org, do, s, accept))$.
The above formula means that if Org is an organization, s a subject, α an action, v a view, cp an object belonging to the $Consent\_preference$ view, and $Consent\_response$ the built-in predicate detailed above then, the $Consent\_context$ holds if there is an object cp, which has the attributes s, v and $NeedConsent(cp)$. When the latter is false, we do not need the consent of the *data_owner* of the object o, else the predicate $Consent\_response$ is needed. By this means, the *dataowner* chooses which view the subject can access.

The Purpose dimension

The purpose is modeled as a user-declared context [49]. Each data owner can create purpose objects to specify the purposes for which access to the personal objects are allowed. The purpose objects belonging to a finite set of PO are grouped and inserted in a purpose view. Purpose values range over the purpose value domain PV (e.g., Medical_research, Inspection). Each purpose object has two attributes:

- *Recipient*, which is a predicate over domains PO x S defining who takes advantage of the declared purpose. That is, if *po* is a purpose object belonging to PO and s is a subject, then *Recipient(po,s)* means that s is the subject who takes advantage of the declared purpose *po*, e.g., in the case of the RFID middleware, the recipient is the client having the URI address.

- *Declared_purpose*, which is a predicate over domains PO x PV, associating a purpose value with the declared purpose object. That is, if *po* is a purpose object and *pv* is a purpose value, then *Declared_purpose(po, pv)* means that *pv* is the purpose value associated with the declared purpose *po*.

By inserting a purpose object *po* in his purpose subview, a data owner declares that another subject (a *Recipient*) will perform some activity in a given context.

Let *do-purpose* denotes the data owner purpose subview associated with his defined objects. The purpose context is specified as follows:
$\forall org \in Org, \forall s \in S, \forall \alpha \in A, \forall o \in O, \forall pv \in PV, \forall po \in PO,$
$Hold(org, s, \alpha, o, userdeclared(pv)) \leftarrow data\_owner(o, do) \wedge Use(org, po, do\text{-}purpose) \wedge Recipient(po, s) \wedge Declared\_purpose(po, pv)$

That is, in organization org, subject s performs action α on object o associated with the data owner do, in the user declared context $user\_declared(pv)$, if there is a purpose object *po* used in the subview *do-purpose* by organization org such that s is the recipient associated with *po* and *pv* is the declared purpose associated with *po*.

Finally, to define a user-declared context two steps have to be realized:

- Specification of the roles who are permitted to declare some given purpose,
- Specification of the roles that are permitted to perform some activities in the associated user-declared context.

The Accuracy dimension

Privacy enforcement requires the use of different levels of accuracy depending on the purpose and the subject requesting the collection of the personal data. This principle is consistent with the privacy directive of collection limitation, cf. Section 5.3.2.

Private objects of each data owner may have different levels of accuracy [8]. We can consider a hierarchy between the root view of a data owner that groups the initially collected objects and their sub-views. These sub-views group the derived objects that have different accuracies. *Sub_view* is a relation over domains *Org* x *V*x *V*. If *org* is an organization, and *v1* and *v2* are views, then *Sub_view(org, v1, v2)* means that in organization *org*, view *v1* is a sub-view of *v2*. Thus, the data owner can define several access policies depending on the different data views of his personal data objects. Foe example, a view of the EPC code or a view of the EPC quantity.

Conclusion

We have seen essential preliminaries concerning the EPCglobal network with a focus on the middleware level. This level is a central point for both data collection and request configurations. We have described the middleware limitations regarding the privacy control and provided an overview on relevant work related to security and privacy in the middleware. Since the act of reading out one or more RFID tags constitutes a data collection, existing privacy laws apply to the communication involving the RFID middleware. Thus, we have introduced the relevant guidelines and regulations based on the Fair Information Practices. Finally, we concluded this chapter by presenting some policy-driven privacy models in the literature, with a focus on the PrivOrBAC model. In the next chapter, we will state the privacy threats related to the reading API of the middleware and propose an approach to integrate the privacy dimensions as part of the EPCglobal specification. The aim is to propose an approach that applies to all applications where critical information are used, specifically, the applications in the context of home healthcare.

CHAPTER 6

Privacy-enhanced Control into the Filtering & Collection Middleware

"I am not a number, I am a free man!" – Patrick McGoohan

Introduction

In the previous chapter, we presented the architecture of an RFID infrastructure, which extends from the reader interface to the entreprise applications. We showed that privacy threats are of real concern in the back-end side of RFID infrastructures, beginning from the middleware level. This level is a key component in which information of each RFID tag is collected and managed, then searched for, and accessed.

In this chapter, we propose a privacy controller module for the RFID middleware in the standardized EPCglobal architecture. We first present the motivation that leads us to introduce a privacy approach into the middleware level. Then, we describe the privacy dimensions that can be enforced in this level. Next, we outline how the existing EPCglobal middleware standard can be modified to satisfy the principles of purpose, consent, and collection limitation (accuracy). Considering a motivation scenario in a healthcare context, we provide a policy-driven approach to enforce the privacy in such a context, using some enhanced contextual concepts of a generic extension of RBAC model. This chapter is ended by providing an integration solution of a privacy module in the middleware, with respect to the EPCglobal specifications.

6.1 Motivation

The middleware sits between the reader and database applications. It is in charge of collecting, filtering and aggregating the requested events from heterogeneous RFID environments,

then compiling them into well-formatted data prior to send them to business application for usability. Holding privacy issues from the lowest possible level of RFID data collection is important. In [41], the authors proposed to handle the dimension of declared purpose in the interface linking readers to tags, which is the lowest level of RFID data collection. In our work, we use the filtering and aggregation properties of the middleware for privacy goals, without interfering with existing standards.

Numerous reasons motivate the support of privacy issues in the RFID middleware. First, treating the privacy in the middleware helps to master the collection configuration and the data view before interpreting and storing it in upper layer applications. This minimizes, as soon as possible, the risk of unauthorized disclosures. In sensitive domains such as healthcare, embedded RFID tags attached to the patient personal devices, tools and medicines can be triggered (after the reader and tag authentication) to reply with their ID and other related information. To illustrate this issue, let us consider two distinct applications of two services in a same hospital wanting to receive data about a patient. The first application is only allowed to view the total tag count depending on the requester purpose, while the second one is allowed to view the identity of tags but only of product "GTIN" when entering in an area of interest. To treat the applications requests, we propose to handle the privacy policy before the events are generated and transmitted to database applications. Second, filtering data for privacy issues at the middleware stage has the advantage to relax the event collection in the middleware, leading to an appropriate adaptation of the queries executed by the reader over the air interface. This allows readers to use efficiently the limited bandwidth, e.g., target only a particular tag population or switch off completely some readers to make the communication bandwidth available to other readers. Finally, as RFID communication protocols support dedicated privacy enhancing features (cf. Part I of the manuscript), the RFID middleware will also need to support their use. For instance, to use the *kill*-command specified in EPC Generation 2 standard, the RFID middleware must provide the appropriate *kill*-passwords to the appropriate reader to apply it to the tag.

As seen in Chapter 5, today RFID interfaces that govern the middleware propose a role based access control as a security approach. However, they fail to address consumer privacy concerns by supporting appropriately known regulations and guidelines [67]. A possible reason for this issue is that most enterprises run the RFID middleware in an internal network [39]. In addition, as explained in [34], the efforts to achieve security and privacy exist in the middleware but not as an architectural incorporation for RFID middlewares; rather as domain specific security rules, implemented up to compliance to specific middlewares. In the sequel, we show how it is possible to integrate a privacy controller on top of a middleware solution to satisfy the basic principles of privacy, without interfering with the existing specifications of the EPCglobal standard. Before delving into the details of our privacy control integrated solution, we show in the following section how to include the privacy dimensions into the ALE interface of the middleware when collecting data.

6.2 Introducing Privacy in the EPCglobal Middleware

It is uncontested that the act of reading out one or more RFID tags constitutes a data collection. Thus, existing privacy laws and regulations apply to the communication involving the RFID middleware that configures the different readers to collect data. That is, the data processing needs to be announced with the purpose and other privacy dimensions, as well as the identity of the data collector. In this section, we outline how the existing EPCglobal middleware standard can be modified to satisfy the dimensions of purpose, consent, and collection limitation (accuracy).

6.2.1 Scenarios for Collecting Data

Various collection needs may differentiate data collectors. The collection could be for local monitoring (i.e., the focus here is to determine whether or not collectors require the serial number of individual tags), or for tracking reasons by tracking multiple occurrences of the same tag across different locations. We identify four distinct collection practices that could be declared by the data collector [41]:

- Anonymous monitoring: collecting state information about the tags in a particular location, without the need to identify the tags by their unique serial number. Examples would be counting tasks for monitoring the number of tags in a certain area or sensor applications as an automatic door opener.
- Local identification: identifications of the tags are collected in order to provide a localization service, such as a smart medicine cabinet that monitors its contents. Even if unique IDs are collected, the application does not attempt to correlate events across different locations.
- Item tracking: collecting information about the location of a tag for the purpose of tracing its movements. This kind of collection potentially enables tracking people through constellations. However, to differentiate between these different intentions, the declaration of *tracking person* should be used in separation, when people are tracked by the tags they carry.
- Person tracking: collecting information about the location of a person. If data collectors collect RFID tag information for this purpose, they need to declare this. Then, it is up to the data owner or legal frameworks to force data collectors to anonymize the tag tracking data so that these data cannot be collected for person tracking.

We are interested in anonymous monitoring and local identification to handle problems of data leakage, which are of importance in scenarios treating private data, e.g., healthcare monitoring systems. Note that the tracking problem was handled in the communication linking readers to tags, by using tag psuedonyms and PRNGs to change them in each request.

6.2.2 Privacy Dimensions in the Middleware

The international community of data protection and privacy commissioners [2] has reported that all the basic principles of data protection and privacy law have to be observed when designing, implementing and using RFID technology. Thus, an RFID middleware should consider these legal principals and guidelines that apply for data collection.

Global System 1 [1] (GS1) has developed the EPC/RFID Privacy Impact Assessment (PIA) tool to help companies uncover the privacy risks and perform their own privacy assessments when implementing their RFID applications [64]. These privacy recommendations have their roots in relevant privacy and data protection laws and regulations previously cited in Chapter 5, such as [67, 135, 44].

Table 6.1 – Privacy Dimensions in the Middleware

Dimension	Middleware support
Collection limitation (Accuracy)	to ensure that only necessary information is collected by holding the accuracy and anonymity specifications in the ECSpec filter
Purpose specification	to announce the valid purpose associated with the request
Explicit consent	to ensure that the data is collected with the knowledge or consent of the data owner.
Security mechanisms	to protect personal data from unauthorized access or disclosure (e.g., encryption, anonymity)

Based on these directives, we have defined, in Section 5.3.2, the seven main privacy dimensions in RFID environments. Some of these principles, such as data owner access or collection relevance for data quality need support primarily in the storage back-end, e.g., with the help of privacy-aware databases. However, the majority of the principles could be supported directly at the middleware data collection level. Table 6.1 lists the privacy dimensions that an extended middleware could support. In this dissertation, we expand the dimensions of purpose, explicit consent, and collection limitation that we call also accuracy.

6.2.3 Privacy Dimensions in the Event Cycle Specification

The information handled by the EPC tag is used to identify assets of an individual or an organization. It could also be useful to know the illness of a person if the EPC identifies a drug. Even by hiding some parts of the tag identity, e.g., by encryption solutions, a partial part of the EPC code can be useful. For example, the combination of an EPC manager identifier and the object class is usually enough to determine the exact kind of object the tag belongs to, which could undermine the consistency of the response. The time when the tag is read, can also present a privacy threat, as in some periods, the holder of the tag does not want to be tracked. Thus, to treat the privacy issues for accurate results, the solution

[1] http://www.gs1.org/

should let all the EPC code to be readable or to entirely filter it. Also, the ECSpec fields for grouping pattern can be a solution to prevent non-authorized disclosures for both tag ID or tag memory content. In addition, the ECSpec specification of time boundaries limits the time for reports notification. Before delving into the details of our proposal, let us present the ECSpec fields, then cross them with the privacy dimensions handled in the middleware level to extract the fields we have to treat to enforce privacy.

Event Cycle Specification (ECSpec)

ECSpec describes an event cycle specification and pre-defines the generated view that could result in some privacy threats (cf. Section 6.2.2). It contains three main parts (cf. Figure 6.1):

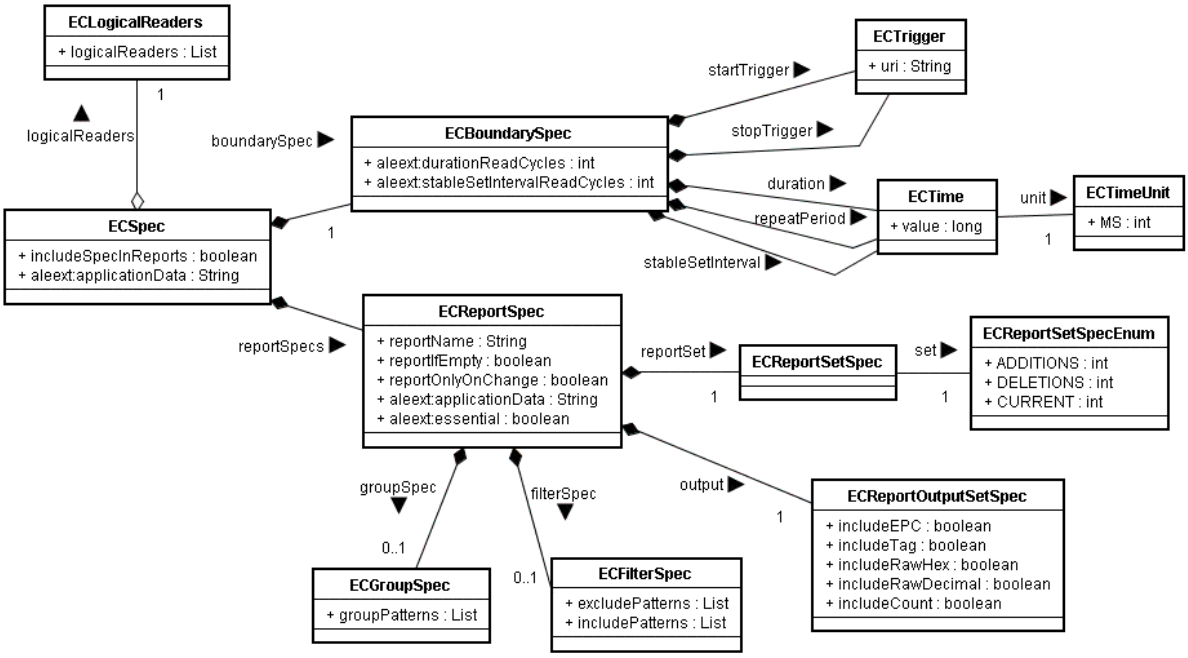


Figure 6.1 – Event Cycle Specification (taken from [138])

- Logical readers or a list of readers (*ECLogicalReaders*). Each member of this list is either a name of a single reader or a composite reader used to read tags.
- Boundary specification (*ECBoundarySpec*). A specification of how the boundaries of event cycles are to be determined. These boundaries specify the start and stop conditions or the duration or the repetition period of the event cycle. The syntax used is: `urn:epcglobal:ale:trigger:rtc:`*period.offset.timezone*. The *period*, in milliseconds, is the time between consecutive triggers occuring within one day. The *offset*, in milliseconds, is the time between the midnight and the first trigger delivered after midnight ($0 \leq$ *offset* $<$ *period*). The *timezone* is the time in which to interpret the specification of the trigger. For example `urn:epcglobal:ale:trigger:rtc:`*60000.0.+01:00*, means that the trigger starts at 00:00 am and repeats every minutes.

- Reports specifications (*ECReportSpec*). Each report specifies one output to be generated from the event cycle and included in the list of reports. Its main fields are:
  - include/exclude patterns (*ECFilterSpec*) to filter what tags are to be included in the final report. A single EPC pattern is a URI-formatted string that denotes a single EPC or a set of EPCs. The syntax used is: `urn:epc:pat:TagEncodingName:Filter.DomainManager.ObjectClass.SerialNumber`. The four fields `Filter`, `DomainManager`, `ObjectClass`, and `SerialNumber` correspond to fields of an EPC tag (i.e., depending on the `TagEncodingName`, some of these fields might not be present[76]). In an EPC pattern, each of the EPC fields cf. Section 2.1.1 can be (1) a decimal integer, meaning that a matching EPC must have this specific value in the corresponding field, (2) an asterisk (*), meaning that a matching EPC might have any value in that field, or (3) a range denoted like `[lo-hi]`, meaning that a matching EPC must have a value between the decimal integers `lo` and `hi`, inclusive. For example, `urn:epc:pat:gid-96:20.*.[6000-8000]` matches any product whose serial number is between 6000 and 8000.
  - grouping pattern (*ECGroupSpec*) defines how filtered tags are grouped for reporting. This parameter separates tags into different groups and is only used when some output format are set. Each pattern in the URI field and its meaning is defined as follows:

    `X` : Create a different group for each distinct value of this field

    `*`: All values of this field belong to the same group

    `Number` : Only EPCs having Number in this field will belong to this group

    `[Lo-Hi]`: Only EPCs whose value for this field falls within the specified range will belong to this group

    Example: `urn:epc:pat:sgtin-64:X.*.*.*` defines groups by filter value (case or pallet).
  - report set (*ECReportSetSpec*) is an enumerated type that specifies the set of tags to be considered for the output. i.e., EPCs read in the *current* event cycle, *additions* or *deletions* from the previous event cycle.
  - output format (*ECReportOutputSpec*) specifies how the final set of EPCs is to be reported. If *includeCount* is true, the report includes also a count of the EPCs in the final set for each group. Each element of this list, when defined, includes four formats: *includeEPC*, *includeTag*, *includeRawHex*, *includeRawDecimal*. If *includeCount* is set to true, the report includes a count of the EPCs in the final set for each group. If both (i.e., element list and *includeCount*) are set to true, each group includes both a format from the list and a count view.

Intersecting privacy dimensions with ECSpec fields

Table 6.2 shows the privacy dimensions that can be treated in each of the ECSpec fields. The configuration of one reader or a list of readers named *logical readers* can directly influence the received reports. The use of a non-authorized reader can result in collecting non-authorized state information (e.g., monitoring the number of tags in a defined area) or providing a localization service (e.g., applied to cases of smart rooms). The use of a list of readers is

mainly viewed as tracking an item or a person, cf. Section 6.2.1. Hence, for privacy reasons, the configuration of this field should also be treated to obtain accurate results. Note that this field is better handled in the ALE LogicalReader API, which deals with logical readers specifications. In this work, the specification of a logical reader is dealt with access control rules (permission or prohibition of access). The specification of *boundary specification* can also raise some privacy threats related to the time or the location of the tag reading. It could be handled by the *temporal consent* dimension when the *duration* field is set (i.e., the *duration* declares the period of time during which the EPC tags are authorized to be read), or by *spatial consent* when *start/stop trigger* is set (i.e., *start/stop trigger* conditions could be used in cases a person is moving from/to an assumed private place, this event could be a trigger to start/stop reading the person tag). Finally, the specification of the *reports* field affects the data view. For accurate results, the process of reading can be treated by including and excluding some EPC tags using the regular expressions [62] or by only reporting a set of tags grouped by an EPC code field when these tags enter in an area of interest. It is

Table 6.2 – ECSpec fields and privacy threats

EC-Spec Field	**Privacy Threats**		**Privacy Dimension**
Logical readers	represents a privacy threat to localize or identify an item. The association of readers can be used to trace the item or a person		accuracy specification / spatial consent
Boundary specification	- Start/Stop Trigger - Duration - Repetition Period	represents a privacy threat if the definition of each reader is not in accordance with the purpose of use and the data owner preference	temporal/spatial consent
Reports	Include/ exclude pattern	represents a privacy threat if some EPCs have not to be included in the final report. This filter fileld is useful for excluding private EPC codes	accuracy specification
	Grouping pattern	is useful to know the quantity of each object type rather than the object's serial number	
	Report set	is a way to represent the collected data. It does not present a privacy issue	No privacy treatment
	Output Format	the format of the set of EPCs to be reported does not present a privacy threat as the filter is performed earlier	

worth noting that the purpose of the request is better specified apart from the ECSpec (cf. Section 6.4.1, for more details).

In the following section, the defined privacy policy model presented in Section 5.4 are allied into a healthcare monitoring scenario using the specification of the F&C middleware.

6.3 Privacy Policy Specification

6.3.1 Motivating Scenario

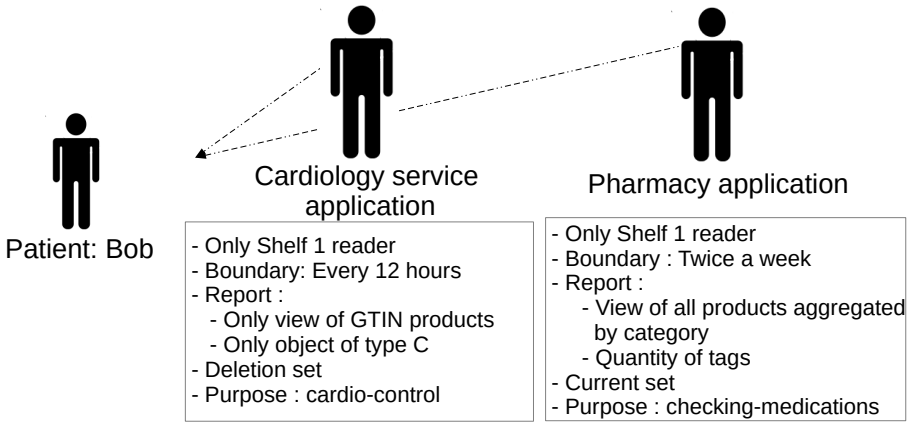


Figure 6.2 – Scenario of a remote monitoring system

This section illustrates a scenario defined in a Hospital Ho for a remote monitoring system. To define the OrBAC organizational policy of Ho, we consider the following entities.

The elderly patient Bob is remotely monitored at home from different services of the Hospital (cf. Figure 6.2). Bob suffers from hypertension and from alzheimer. The medications he daily takes are put in his medication shelf and are identified with EPC codes i.e., every one pill is related to one EPC code. The nurse of the cardiology service has to monitor Bob twice a day. Every time Bob takes his hypertension medication from the shelf, the total number of pills decreases. The nurse has not to track the tags that are related to the alzheimer disease. This latter information is considered as private with respect to the patient preferences. In the other side, the Pharmacy application has to monitor the medication shelf twice a week, to check if it contains the required quantity of medications. Thus, it only needs a quantity information about the medications rather than their entire EPC code.

Based on these assumptions, the nurse of cardiology service is only interested in objects of type C, i.e., the control is done over the field *object* of the EPC code (cf. Figure 2.1), describing the reference of the hypertension medication. The ECSpec is configured to only consider *deletions_set* of tags, i.e., tags deleted from the previous event cycle. Finally, the pharmacist is only interested in tags quantity depending on their manager and object codes.

Figure 6.2 shows the required configuration of the ECSpec with a privacy point of view. As we will argue in Section 6.4.1, the purposes are entered apart from ECSpec fields.

6.3.2 The OrBAC Specification

The privacy rules corresponding to the motivating scenario are expressed in the OrBAC model as follows. Note that the rules of the two applications are defined separately.

The cardiology nurse application

The nurse related to the cardiology application is only permitted to receive data of patients suffering from hypertension, with the context *Consent_nurse* and the view *TagC_type* in the user-declared context *patient_nurse*.

- The user declared purpose and the accuracy rules are described in two steps:
  1. The *dataowner* who is the patient or another person to whom the patient has delegated the right to control the access to his data (e.g., an authorized relative) defines the context *cardio_control*.
  2. In hospital Ho, a user empowered in the role *nurse* is permitted to have an access to the generated reports containing only tags of type C in the declared context *cardio_control*. The nurse has no access to other tagged objects.

In the ALE middleware, the subscription request requires two subjects who are involved in the process of context declaration: the subject who is requesting the ALE middleware and the subject who takes advantage of this request. Our model considers the "real" recipient which is the *nurse*, independently of the one who has activated the subscription (i.e., it can be the patient nurse, or the doctor). The above described rules are modeled as follows:

- First, we specify that the patient (or any empowered person if the patient is unable to do so) who is the *dataowner* is permitted to define the purpose *cardio_control* that applies to one of his nurses. This is done by the following permission rule:

  $Permission(Ho, patient, declare, cardio\_control, patient\_nurse)$,

  where *cardio_control* is a purpose associated with some objects and *patient_nurse* is a context defined as follows:

  $\forall Ho \in Org, \forall s \in S, \forall \alpha \in A, \forall po \in PO$,

  (1) $Use(Ho, po, cardio\_control) \leftarrow Use(Ho, po, do\text{-}purpose) \wedge Declared(po, cardio\_control)$
  (2) $Hold(Ho, s, \alpha, po, patient\_nurse)) \leftarrow Use(Ho, po, \text{do}-\text{purpose}) \wedge PatientID(po, s) \wedge Recipient(po, s') \wedge Nurse(s', s)$

That is, (1) the hospital Ho uses the purpose object po in the subview *do-purpose* (*do-purpose* is the *dataowner* subview of the view $Purpose$ associated with his defined objects), such that *cardio_control* is the declared purpose associated with po. In addition, (2) in hospital Ho, a subject s performs an action α on the object o associated with the patient s in the context *patient_nurse* if there is a purpose object po used in the subview *do-purpose* by Ho such that s' is the recipient associated with po (represented by the application-dependent predicate $Recipient(po, s')$) and s' is the nurse of the patient s (represented by the application-dependent predicate $Nurse(s', s)$).

- Second, we specify that subjects empowered in the role $Nurse$ are permitted to collect objects belonging to the view $TagC\_type$ in the user-declared context *cardio_control*, as follows: $Permission(Ho, Nurse, collect, TagC\_type, cardio\_control)$

- The consent rule specification:

  $Rule_1$ = *Permission(Ho, NurseApp, Subscribe, TagC_type, Consent_nurse)* where the context *Consent_nurse* is specified as:

  $Rule_{consent\_1}$: $\forall$ *Ho* $\in$ *Org*, $\forall$ *s, do* $\in$ *S*, $\forall\alpha \in A$, $\forall$ *o* $\in$ *O*, $\forall$ *cp* $\in$ *O*, *Hold(Ho, s,* α*, o, Consent_nurse)* $\leftarrow$ *Use(Ho, cp, Consent_preference)* $\wedge$ *Recipient(cp,s)* $\wedge$ *Target(cp, o)* $\wedge$ *Dataowner(cp, do)* $\wedge$ ($\neg$ *NeedConsent(cp)* $\vee$ *Consent_response(Ho, do, s, cp, accept))* $\wedge$ *ReadingPeriod-Nurse* .

  That is, if *Ho* is the organization, *s* is a subject performing an action α on the object *o*, *cp* is an object belonging to the *Consent_preference* view of the data owner *do* and *Consent_response* the built-in predicate detailed in Section 5.4.2, then, the *Consent_nurse* context holds if there is an object *cp* belonging to the *Consent_preference* view which has the attributes *s* and *NeedConsent(cp)*. The data owner response is modeled by a built-in predicate $Consent\_response$, when the consent is needed ($NeedConsent = true$). The reading period is a time constraint defining the period of reading the tags that is related to the nurse application. The period is equal to 12 hours, as defined in the motivating scenario (cf. Section 6.3.1).

The pharmacy application

The pharmacy application is only permitted to receive data about patients, with the context *Consent_pharmacist* and the view *AllTagNumber* in the user-declared context *checking_medications*:

- The user declared purpose and the accuracy rules are described in two steps:

  1. The *dataowner* who is the patient or another person to whom the patient has delegated the right to control the access to his data (e.g., an authorized relative) defines the context *cardio_control*.

2. In hospital Ho, a user empowered in the role *pharmacist* is permitted to have an access to the generated reports containing only tags number in the declared context *checking_medications*. The pharmacist has not to know the identification of the EPC tags.

In the ALE middleware, the subscription request requires two subjects who are involved in the process of context declaration: the subject who is requesting the ALE middleware and the subject who takes advantage of this request. As with the nurse application, our model considers the "real" recipient which is the *pharmacist*, independently of the one who has activated the subscription. The above described rules are modeled as follows:

- First, we specify that the patient (or any empowered person if the patient is unable to do so) who is the *dataowner* is permitted to define the purpose *checking_medications* that applies to one of his pharmacists. This is done by the following permission rule:

  $Permission(Ho, patient, declare$,checking_medications, $patient\_pharmacist)$,

  where *checking_medications* is a purpose associated with some objects and *patient_pharmacist* is a context defined as follows:

  $\forall Ho \in Org, \forall s \in S, \forall \alpha \in A, \forall po \in PO$,

  (1) $Use(Ho, po, checking\_medications) \leftarrow Use(Ho, po, do\text{-}purpose) \wedge Declared(po,$-*checking_medications*)
  (2) $Hold(Ho, s, \alpha, po, patient\_pharmacist) \leftarrow Use(Ho, po, \text{do}-\text{purpose}) \wedge Patient\text{-}ID(po, s) \wedge Recipient(po, s') \wedge Nurse(s', s)$

  That is, (1) the hospital Ho uses the purpose object po in the subview *do-purpose* (*do-purpose* is the *dataowner* subview of the view $Purpose$ associated with his defined objects), such that *checking_medications* is the declared purpose associated with po. In addition, (2) in hospital Ho, a subject s performs an action α on the object o associated with the patient s in the context *checking_medications* if there is a purpose object po used in the subview *do-purpose* by Ho such that s' is the recipient associated with po (represented by the application-dependent predicate $Recipient(po, s')$) and s' is the nurse of the patient s (represented by the application-dependent predicate $Nurse(s', s)$).

- Second, we specify that subjects empowered in the role $Pharmacist$ are permitted to collect objects belonging to the view $AllTagNumber$ in the user-declared context *checking_medications*, as follows:

  $Permission(Ho, Pharmasist, collect, AllTagNumber, checking\_medications)$

- The consent rule specification:

  $Rule_2$ = *Permission(Ho, PharmacyApp, Subscribe, AllTagNumber, Consent_pharmacist)* where the context *Consent_pharmacist* is specified as:

$Rule_{consent\_2}$: $\forall$ *Ho* $\in$ *Org*, $\forall$ *s*, *do* $\in$ *S*, $\forall\alpha \in A$, $\forall$ *o* $\in$ *O*, $\forall$ *cp* $\in$ *O*, *Hold(Ho, s,* α*, o, Consent_pharmacist)* $\leftarrow$ *Use(Ho, cp, Consent_preference)* $\wedge$ *Recipient(cp,s)* $\wedge$ *Target(cp, o)* $\wedge$ *Dataowner(cp, do)* $\wedge$ *(*$\neg$ *NeedConsent(cp)* $\vee$ *Consent_response(Ho, do, s, cp, accept))* $\wedge$ *ReadingPeriod-Pharmacist.*

That is, if *Ho* is the organization, *s* is a subject performing an action α on the object *o*, *cp* is an object belonging to the *Consent_preference* view of the data owner *do* and *Consent_response* the built-in predicate detailed in Section 5.4.2, then, the *Consent_pharmacist* context holds if there is an object *cp* belonging to the *Consent_preference* view which has the attributes *s* and *NeedConsent(cp)*. The data owner response is modeled by a built-in predicate *Consent_response*, when the consent is needed ($NeedConsent = true$). The reading period is a time constraint defining the period of reading the tags that is related to the nurse application. The period is equal to 84 hours, as defined in the motivating scenario (cf. Section 6.3.1).

6.4 Privacy-enhanced Filtering & Collection Middleware

The integration of a privacy controller module in the EPCglobal F&C middleware, is depicted in Figure 6.3. Relying on the ALE interactions for reading tags (cf. Chapter 5) and the privacy dimensions required at this level (cf. Section 6.1), we consider that the subscription to an ECSpec (or the other request modes) is the action concerned with the privacy issues. In fact, the collection of data is only triggered by this action, since the defined ECSpec is not executed until the subscription is performed. The other activities (e.g., the definition of an ECSpec) could be controlled by classical access control policies.

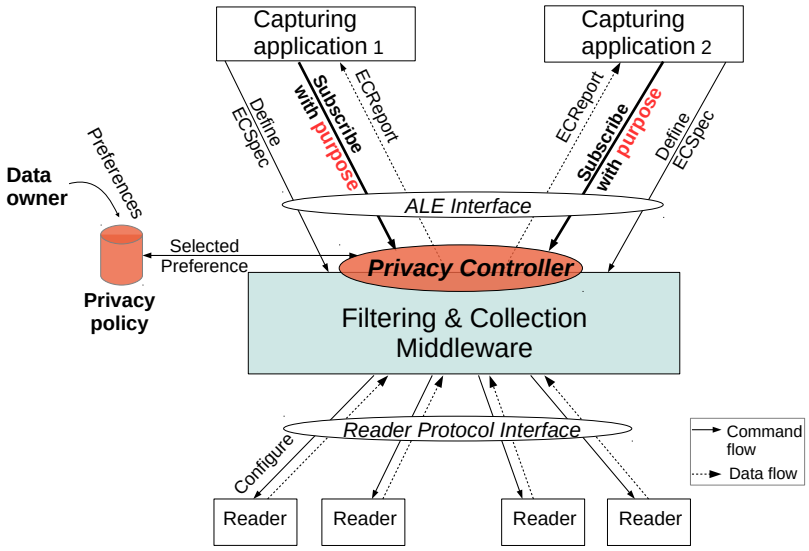


Figure 6.3 – EPCglobal middleware with a privacy controller module

Regarding the *purpose* declaration, it can either be added to the ECSpec content or as a parameter with the *subscribe* request. We add the *purpose* to the *subscribe* request for the following reasons. First, this way, the ECSpec can be used by many capturing applications and with possibly different purposes. Second, adding the purpose parameter to the subscription request avoids distorting the ECSpec and leaves it consistent with the EPCglobal standard specification.

When triggered with the client subscription, the privacy controller calls the privacy policy database, where data owner preferences are stored. It compares these entered preferences and the content of the ECSpec request and should be able to decide whether to grant or deny the collection of the specified data or to update the ECSpec. In the sequel, we describe the privacy controller activities when triggered by a subscription request.

6.4.1 Privacy Controller Activities

To subscribe to an event cycle and receive related reports, the client calls, via the ALE interface, an entity responsible for reports generation, referred here as *CentralEntity*, (cf. Figure 6.4). We define the *PrivacyController* as the entity handling the compliance of the subscription requests with the predefined privacy policy.

Figure 6.4 depicts the relation between the two defined entities (i.e., *CentralEntity* and *PrivacyController*). The *CentralEntity* instantiates the *PrivacyController* with the specification ECSpec, the recipient address URI and the purpose to verify the properties of the request. The *PrivacyController* outputs, after calling the privacy policy a Grant or Deny of access to the specified data, otherwise updates the ECSpec for accurate results.

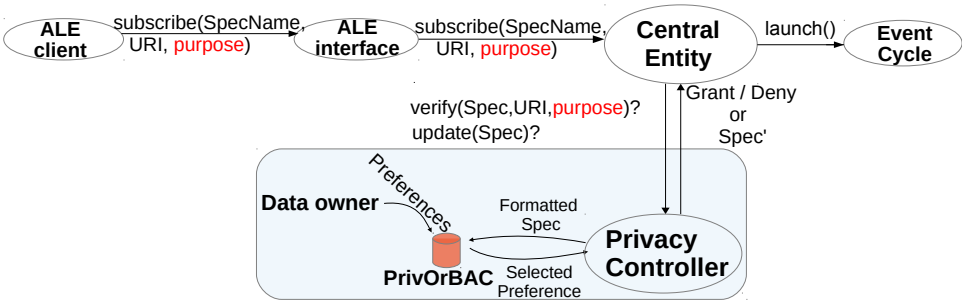


Figure 6.4 – Privacy controller activities

Regarding the privacy preferences, the data owner specifies them using the PrivOrBAC model [8] depending on the privacy dimensions enforced in the system. These preferences are introduced in a predefined ontology. As in Figure 6.5, each data owner $preference$ is associated with a *recipient* address and a *purpose* on which the preference is defined. This preference turns around a set of targeted objects (*targets*) that the *recipient* aims to access. A *decision* attribute, related to each *target*, defines whether the data owner gives his consent

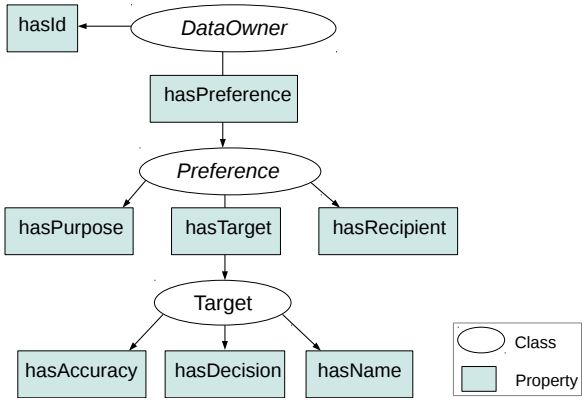


Figure 6.5 – Privacy preferences ontology

to access the data or not. In the positive case, the *decision* attribute designates the accuracy in which the data will be disclosed. Note that the final decision has to be in accordance with general privacy rules of relevant regulations, which are initially specified in PrivOrBAC.

As an illustration, we apply our methods to the motivating example, presented in Section 6.3.1. Figure 6.6 represents the preferences of the data owner as implemented in PrivOrBAC. Recall that for the cardiology nurse, the patient Bob decides to disclose only his medications of type C in clear (i.e., the entire EPC code) for the declared purpose $Cardio\_control$. Regarding the pharmacist application, the patient Bob allows the collection of all the tags but with a numbering view depending on their EPC manufacturer field. In the two cases, the purpose should be declared along with the subscription.

In more details, according to the privacy policy, if the ECSpec of the nurse application, only contains the type C *target*, and the specified duration is in accordance with the time constraints, the collection of the data is authorized. If the Nurse requests for more EPC objects as *targets*, these latter are updated in accordance with the data owner preference, by changing the defined pattern for this field. e.g., the pattern entered in the ECSpec : `urn:epc:pat:sgtin-96:0000389.*.*`, is replaced by

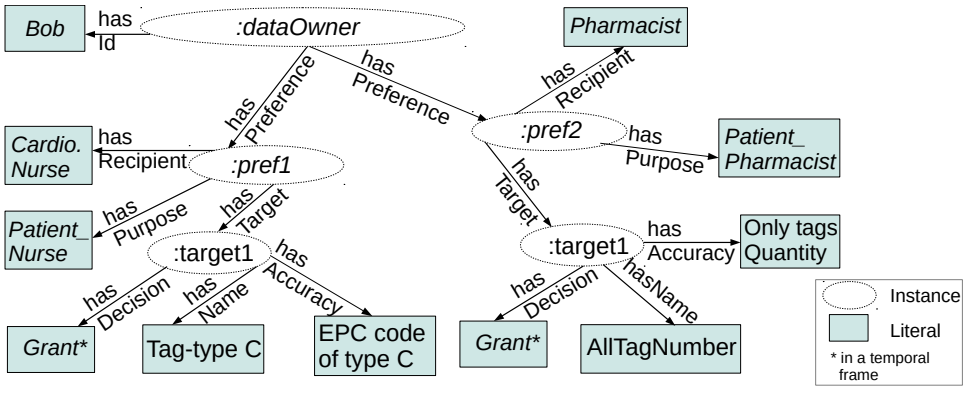


Figure 6.6 – Privacy preferences in the motivating scenario

Algorithm 3 Subscribe to an ECSpec

Input: ECSpec `spec`, purpose `purp`, recipient `URI`

```
1: NewSubscriber ← URI
2: if (PrivacyController.verify(spec, URI, purp)) then
3:    ECSpec spec' ← PrivacyController.update(spec)
4:    if spec' != Null then
5:       spec ← spec'
6:       SubscribersLIST.put(URI)
7:    else
8:       ALEMessage ← No update to be performed
9:    end if
10: else
11:    ALEMessage ← The URI has not the right to subscribe to ECSpec with the purpose purp
12: end if
```

`urn:epc:pat:sgtin-96:0000389.1234.*`, to specifically designate an object of type `1234`. The same update is done over the duration field. Note that the representation of the EPC objects is identically defined in both PrivOrBAC framework and ECSpec.

6.4.2 Privacy Control for Subscriptions

The *PrivacyController* is defined as an independent entity that *CentralEntity* calls for new subscriptions. Our *PrivacyController* entity uses two main methods: *Verify* to check the subscription parameters, i.e., the purpose, the ECSpec name and the logical reader field and *Update* to change the ECSpec, when needed.

The *Verify* method returns a boolean:

- *False* when the entered parameters are not correct, e.g., the specified purpose and logical reader are not adequate for this access,
- *True* when the subscription to the ECSpec is entirely allowed (e.g., already existing in a defined database) or when it is allowed but with possibly, further filtering for privacy reasons, e.g., more EPC *targets* than allowed are specified. In this latter case, the *Update* method is called.

The *Update* method updates the ECSpec content with the corresponding data owner preferences. It compares the ECSpec fields entered by the recipient and the preferences of the data owner, then updates each field of the ECSpec, except the logical readers, which update is rather handled by the logical reader API, cf., Section 6.2.3. At the end, the *Update* method checks for the correctness of the resulted ECSpec `spec'`. In the case `spec'` is not correct, the method outputs `Null`, no changes are performed on the ECSpec, and an empty report is generated.

Algorithm 3 shows the *subscribe* method that could be included in the *CentralEntity*. Steps 2 and 3 verify the existence of an already subscribed recipient, identified by its URI address with the same ECSpec. Step 5 verifies the permission of the recipient to subscribe to the report notifications with the entered ECSpec and purpose. In the positive case (Step 3), a call to the *Update* method is performed. If the output of the *update* method is True (cf. Step 4), then the original ECSpec is required to be updated with a new filtered ECSpec. Step 6 adds the address URI to the list of validated subscribers in case of permission. Note that for checking the correctness, some general criteria should be satisfied by any updating algorithm, to generate sound, maximum and secure results (cf. Appendix C, for more details). As an example, if the period of reading tags specified by the subscriber is different from the period specified by the data owner, the minimum of the two values is considered. Thus, the subscription is not rejected and the criteria of maximality is applied.

Conclusion

We have shown the importance of introducing privacy dimensions at an early stage of RFID collection and data processing to prevent unintentional disclosures of sensitive information. Our goal is to ensure that the communication line that extends from readers to middleware and entreprise applications responds to privacy requirement of relevant regulations. We have addressed some privacy concerns of the EPCglobal technology and provided a policy-driven approach to enforce privacy in such a technology with the dimensions of declared purpose, accuracy and explicit consent. This chapter is concluded by defining the new modeled middleware of EPCglobal network, which includes the privacy module responsible for enforcing the privacy in the system. In the next chapter, we will provide a proof of concept prototype to show the feasibility of our approach using the Fosstrack platform.

CHAPTER 7

Platform and Implementation

"The strongest arguments prove nothing so long as the conclusions are not verified by experience. Experimental science is the queen of sciences and the goal of all speculation." – Roger Bacon

Introduction

In the previous chapter, we provided a privacy control model to integrate the required privacy dimensions in the middleware level of the EPCglobal network. To show the feasability of our privacy-enhanced model in the middleware, we provide in this chapter a proof of concept showcasing its applicability into the filtering and collection (F&C) middleware of the Fosstrak platform. Fosstrak is an open-source implementation of the EPCglobal specifications, providing an EPCglobal-certified EPCIS Repository. First, an introduction of the Fosstrak platform, its modules and implementation details is given. The aim is to locate the key point, where event cycle specifications are evaluated to introduce our privacy controller module. Then, a concrete implementation of the privacy controller is shown. After presenting our modifications, we provide a testing scenario and associated results to show the effectiveness of our approach. To end this chapter, an evaluation of our integrated module is given, in terms of execution time.

7.1 The Fosstrak Platform

The EPCglobal organization has defined a group of standard interfaces but does not specify a particular implementation strategy for the different roles in the EPCglobal network. Fosstrak (ex Accada) [69] is an open-source RFID software platform implementing the standardised roles and interfaces published by EPCglobal. More specifcally, it provides an EPCglobal-certified EPCIS Repository as well as Query and Capture clients. Additionally, Fosstrak features a number of extensions that address some of the challenges of RFID middleware designs [40]. The Fosstrak platform consists of three separate modules: the reader module, the filtering and collection middleware module, and the EPCIS module, which deals with interpretation of the captured RFID data in an application context. Each of the three modules implement the corresponding roles in the EPCglobal network (described in Figure 5.1 of

Chapter 5), as well as the interface specifications of the reader protocol, the ALE (version 1.1) and the EPCIS Query and Capture interfaces. In Fosstrak, the interfaces to request the ALE and readers are modeled in the WebServices Description Language[1] (WSDL). To communicate with RFID readers, the Fosstrak ALE middleware uses the EPCglobal LLRP [63] (Low Level Reader Protocol). For readers that do not support LLRP, the ALE middleware uses the Fosstrak Hardware Abstraction Layer (HAL).

We frame our work in the middleware module, the key location where the configuration of events to be collected is performed and subsequently, related reports are generated. Nevertheless, to test the whole process of the filtering and collection of EPC events, all roles and interfaces, directly related to the EPCglobal middleware (cf. Figure 6.3 in Chapter 6), should be well configured and connected.

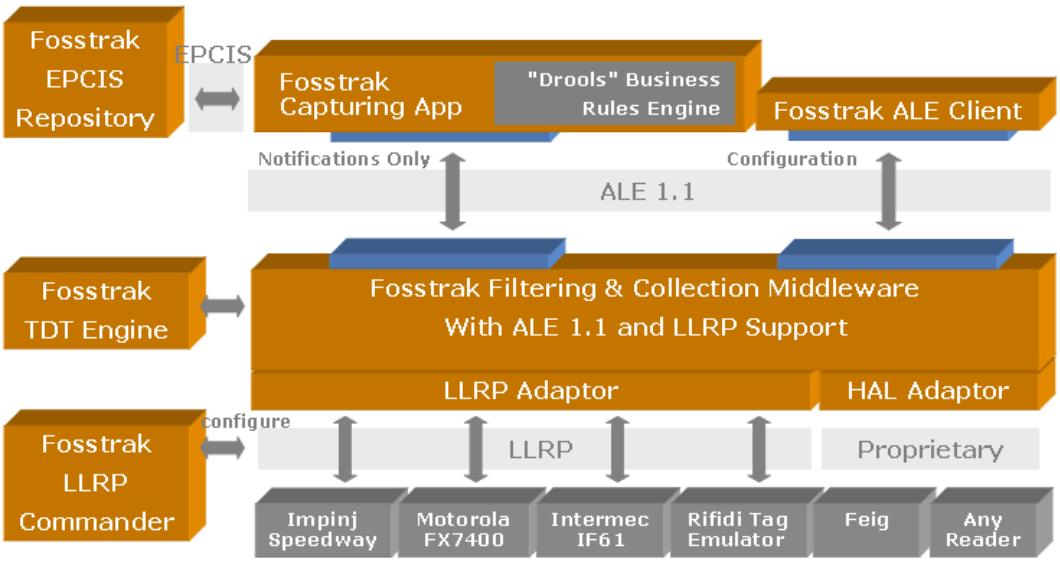


Figure 7.1 – Fosstrak platform (taken from [69])

7.1.1 Fosstrak Middleware Modules

The Fosstrak ALE middleware comprises two modules: the F&C server and a standalone client or a web-based client to configure the F&C servers implementing the ALE 1.1 interface:

- The F&C server is an implementation offering the features of the F&C middleware role as defined by EPCglobal's ALE standard. It currently supports the ALE Logical Reader API that is used to define logical readers through Logical Reader Specifications (LRSpec) and the ALE Reading API that is used to define the filtering and collection behavior using ECSpecs.
- The client that could be a web-based or a standalone Java client. The first is a Java web application, compiled in Web Application Archive files, that allows developers to

[1] http://www.w3.org/TR/wsdl

use their web browser to define ECSpecs, endpoints as well as LRSpecs and submit them to a F&C server. The second client is a a graphical user interface to define ECSpecs and LRSpecs on a particular filtering and collection server, providing the same functionality as the web based client.

In the sequel, we use the F&C server to add our privacy controller module and the web-based client to test our extension. The modification are performed on the Fosstrak middleware of the version 1.2.0, implemented in Java language. We begin by exploring the event cycle implementation in Fosstrak, then define the reports generation entity, which provides the key place where the privacy module should be integrated.

7.1.2 Fosstrak Event Cycle and Reports Generators

Recall that according to the EPCglobal standard, an event cycle is the smallest unit of interaction between the ALE client and the ALE implementation through the ALE reading API. An event cycle is an interval of time during which tags are read. In Fosstrak, the event cycle is implemented as a thread named *EventCycle*. After its creation, the thread waits on the *EventCycle* monitor, cf. Listing 7.1.

```
1  /** This method is the main loop of
       the event cycle in which the
       tags will be collected.
2  * At the end the reports will be
       generated and the subscribers
       will be notified.*/
3
4  ...
5
6  public void run() {
7      ...
8      ...
9
10 // Running will be set by the
       ReportsGenerator when the
       EventCycle has a subscriber
11 if (!running) {
12   synchronized (this) {
13     try {
14       this.wait();
15     } catch (InterruptedException e
           ) {
16       LOG.info("eventcycle got
             interrupted");
17       return;
18     }   }
19 }}
```

Listing 7.1 – Portion of the *EventCycle* code

```
21 /** This method contains the main loop
       of the reports generator.
22 * Here the event cycles will be
       generated and started. */
23
24 public void run() {
25       ...
26 // while state is REQUESTED start every
       repeatPeriod a new EventCycle
27   while (isStateRequested()) {
28     if (eventCycle == null) {
29       LOG.error("eventCycle is null");
30     } else {
31     eventCycle.launch();
32     }
33     try {
34       synchronized (state) {
35         state.wait(repeatPeriodValue);
36       }
37 // wait for the event cycle to finish
38       eventCycle.join();
39     } catch (InterruptedException e) {
40     LOG.debug("caught interrupted
         exception - leaving reports
         generator.");
41     return;
42     }}}
```

Listing 7.2 – Portion of the *ReportsGenerator* code

When there are subscribers to the *EventCycle*, the *ReportsGenerator* entity launches the *EventCycle* through the *launch* method. Listings 7.1 and 7.2 show the relation between *EventCycle* and *ReportsGenerator* implementations.

The *EventCycle* runs as long as the specified duration in the corresponding ECSpec field expires, (cf. Section 6.2.3, for Event Cycle Specification fields). Then, the reports are generated and sent through the *ReportsGenerator* to the subscribers.

Event Cycle creation and flow

An *EventCycle* is constructed according to an event cycle specification, ECSpec. The ECSpec specifies several parameters (i.e., the time interval during which tags are read, the readers where tags shall be collected). Whenever a client defines a new event cycle (ECSpec) through the ALE interface, a new *ReportsGenerator* will be created along an *EventCycle*. Upon creation, the *EventCycle* aquires the readers from the logical reader API and registers as an Observer. As soon as a client subscribes to the *EventCycle*, the *ReportsGenerator* starts the *EventCycle*. Then, tags are read and returned to the *ReportsGenerator*.

In point of fact, a client application does not subscribe to an EventCycle to receive its specified events, rather the subscription is done through the associated *ReportsGenerator*. The *ReportsGenerator* is the central entity ensuring that the EventCycle is started/stopped and that subscribers receive the resulting tags. Therefore, the *ReportsGenerator* acts as a gateway to the *EventCycle* for clients.

7.1.3 Sequence Diagram of Reports Generation

As observed, the *ReportsGenerator* plays a central role in the process of report generation related to event cycles. The sequence diagram in Figure 7.2 visualizes the behavior of both event cycle and report generator entities.

Assuming that the client and the recipient of the report represent the same entity, the following steps describe the message flow:

1: A client defines a new event cycle (spec), named SpecName through the ALE.
1.1: The ALE checks if this event cycle already exists. In the present case, there is no such event cycle. Hence, the ALE creates a new *ReportsGenerator*.
1.1.1: The *ReportsGenerator* creates a new *EventCycle*.
1.1.1.1/2/3: The *EventCycle* retrieves the readers specified in the ECSpec from the LogicalReaderManager.
2: The Client subscribes for an *EventCycle*.
2.1: The ALE subscribes the Client to the *ReportsGenerator*.
2.1.1: The *ReportsGenerator* starts the *EventCycle*. The *EventCycle* now processes tags from the readers.

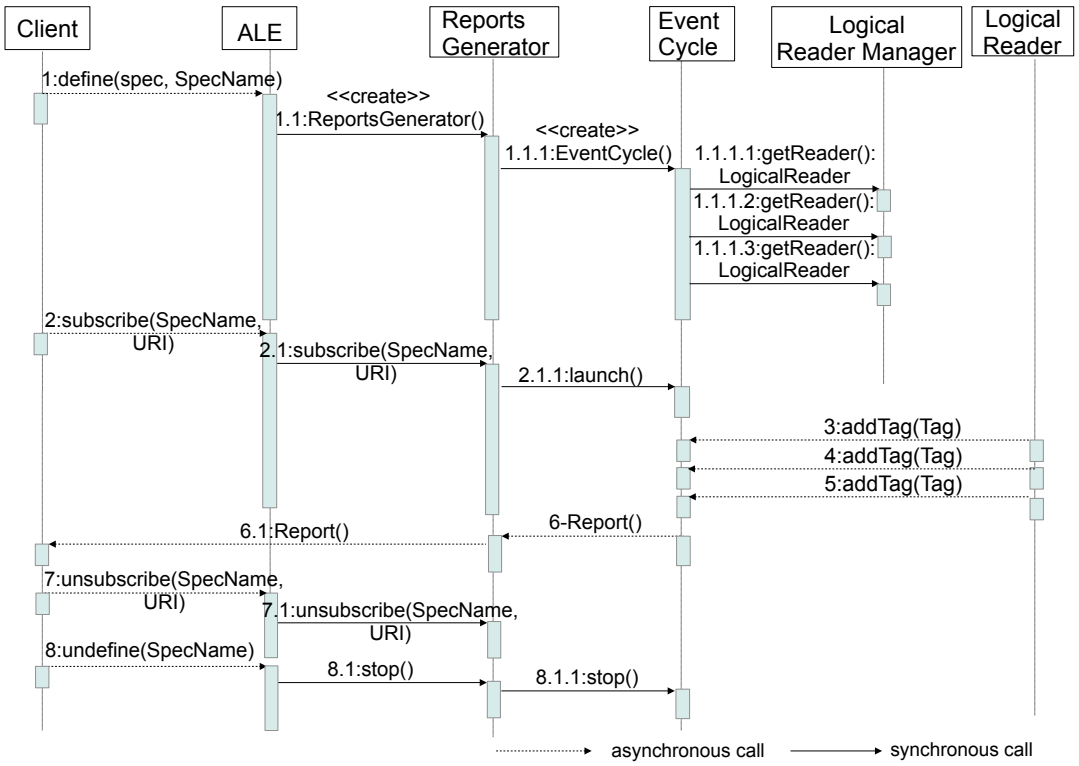


Figure 7.2 – Sequence diagram for subscription

3/4/5: The LogicalReader(s) add Tags to the *EventCycle*.
6: When the *EventCycle* reaches its boundaries the reports are generated and sent back to the Client (through the *ReportsGenerator*).
7: The Client wants to unsubscribe from an *EventCycle*.
7.1: The ALE calls the unsubscribe method on the corresponding *ReportsGenerator*.
8: The Client undefines an *EventCycle*.
8.1: The ALE stops the corresponding *ReportsGenerator*.
8.2: The *ReportsGenerator* stops the *EventCycle*

7.2 Privacy Controller Implementation in Fosstrak

The *PrivacyController* module written in Java, is added to the version 1.2.0 of the Fosstrak middleware. It is defined in the Fosstrak code as an independent class that *ReportsGenerator* calls for new subscriptions.

7.2.1 Modifications in the Fosstrak Server and Client

We have modified the Fosstrak filtering and collection server (*fc-server*) and changed the Web-based client to support the *purpose* attribute as an entry along with the ECSpec, and the URI address. In the *fc-server*, we have mainly changed the *ReportsGenerator* implementation and all the classes related to it (in Appendix D, Figure D.1 shows a tree structure of the main modifications done on the original version of *fc-server*). More specifically, we have added the *PrivacyController* class and its methods into the Fosstrak filtering and collection server and changed the `subscribe` method of the *ReportsGenerator* class. Listing 7.3 depicts a snapshot of the `subscribe` method of the modified *ReportsGenerator* class.

```
1  //Constructor validates the ec specification and sets some parameters.
2  public ReportsGeneratorImpl(String name, ECSpec spec, ECSpecValidator validator,
       ECReportsHelper reportsHelper) throws ECSpecValidationException,
       ImplementationException {
3      ...
4      // set name and spec
5      this.name = name;
6      this.spec = spec;
7      ...
8  // This method subscribes a notification uri of a subscriber to this report
       generator.
9    @Override
10   public void subscribe(String notificationURI, String purpose) throws
         DuplicateSubscriptionException, InvalidURIException, NoRightException {
11     System.out.println("purpose: " + purpose);
12     Subscriber uri = new Subscriber(notificationURI);
13     if (subscribers.containsKey(notificationURI)) {
14       throw new DuplicateSubscriptionException(String.format("the URI is already
             subscribed on this specification %s, %s", name, uri));
15     } else {
16       if (PrivacyController.verify(name, spec, notificationURI, purpose)){
17         ECSpec result = PrivacyController.update(spec);
18         if (result != null)
19           {
20             spec = result;
21           subscribers.put(notificationURI, uri);
22           LOG.debug("NotificationURI '" + notificationURI + "' subscribed to spec '
                 " + name + "'." +"with purpose" + purpose);
23           if (isStateUnRequested())
24             {
25             setState(ReportsGeneratorState.REQUESTED);
26             }
27           }
28       } else {
29         throw new NoRightException(String.format("No right to subscribe %s, %s, %s"
               , notificationURI, name, purpose));
30         }
31     }
32   }
```

Listing 7.3 – Modified *subscribe* method into the *ReportsGenerator*

We have also changed the *fc-common* directory of the Fosstrak middleware (cf. Figure D.2 in Appendix D) to integrate exception treatments in relation with the right to collect the data. To take into account these changes, a compilation of the server as a Web application ARchive (WAR) file is needed. Figure 7.3 shows the new Fosstrak web-based client interface where the purpose attribute is added as a required field to be taken into account for the subscription.

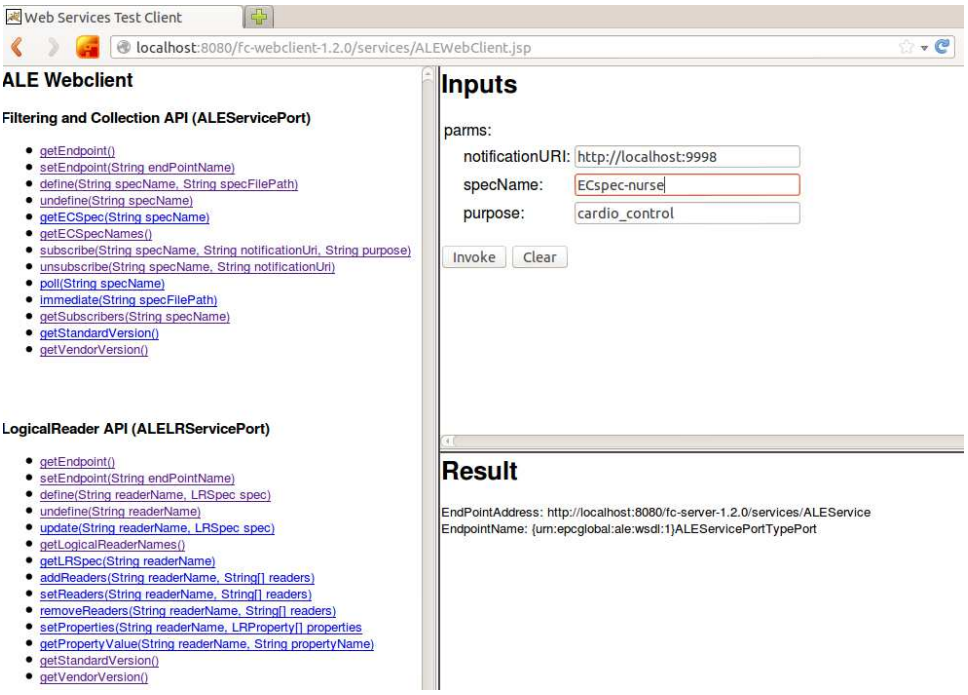


Figure 7.3 – Web-based client interface

7.2.2 Testing Scenario in a New Sequence Diagram of Reports Generation

As stated before, to test the whole process of the filtering and collection of EPC events, all roles and interfaces directly related to the EPCglobal middleware (cf. Figure 7.1) should be well configured and connected.

Implementation details: To ensure the communication between the F&C middleware and the different readers, we consider the EPCglobal Low Level Reader Protocol [63] (LLRP). On the other side, we use a reader that supports the same protocol (LLRP). We use the LLRPCommander tool [2] that provides a number of features to help configuring and managing LLRP-compliant RFID readers. To simulate LLRP readers and generate tag data, we use the reader emulator Rifidi[3]. Finally, to simulate ALE clients (i.e., capturing application), we use the F&C Web Client application compiled in Java web application, which comes with the Fosstrak platform (cf. Section 7.1.1). Regarding the management of the data owners

[2]http://code.google.com/p/fosstrak/wiki/LlrpMain
[3]http://www.transcends.co/community

preferences, the PrivOrBAC policy is stored in a dedicated server and user's preferences are accessed via a web service [139], using the REpresentational State Transfer (REST) architecture. For the configuration steps, please refer to the Appendix D.

In order to generate the event cycle reports, an ALE Client should subscribe to a set of events specified in its entered event cycle specification ECSpec. To visualize the new messages flow related to the subscription activity, Figure 7.4 depicts a sequence diagram including the privacy controller module we have integrated into the Fosstrak platform. We assume that the client and the recipient of the report represent the same entity. The goal is to receive the reports corresponding to the client specification, when the context holds with the rules defined in the privacy policy. In the sequel, we apply the motivating scenario when describing the flow of messages in the Fosstrak platform

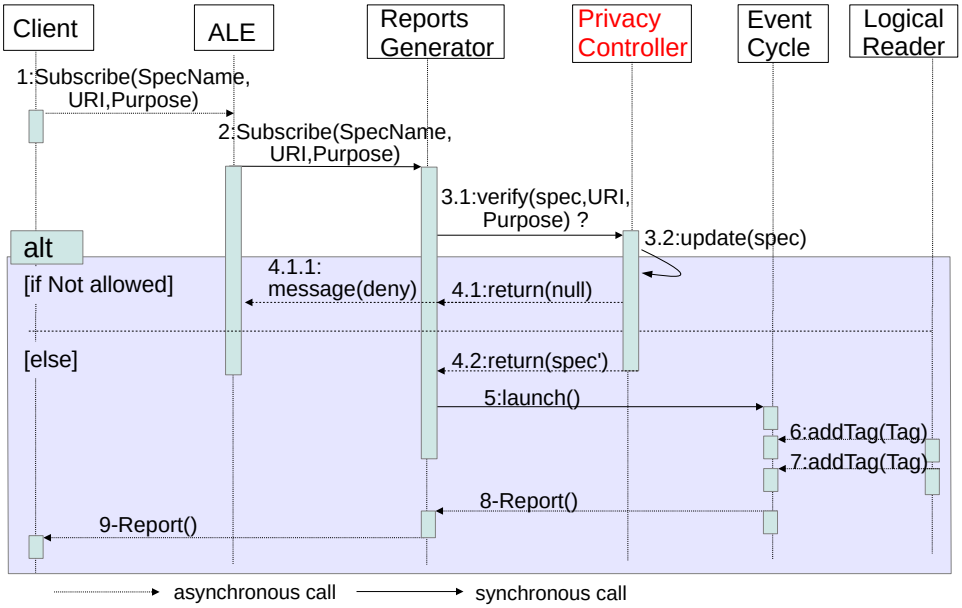


Figure 7.4 – Sequence diagram related to subscriptions

1: The Client subscribes to an *EventCycle*, by entering the ECSpec (assumed to be already created), the recipient address (URI), and the purpose (from a purpose list). Figures 7.5 and 7.6, respectively illustrate the ECSpecs of the nurse and the pharmacist applications rendered into XML files. In the case of the nurse application, the EPC code partition `0000389` refers to the manager code for a given corporation, and the partition `000162` denotes the product code for hypertension medication (Type C), where `GTIN-96` tag encoding is used. The *reportSet* field is set to *DELETIONS*, to only detect deleted tags from the *shelf1* reader. We assume that the nurse is only interested in the `epc` code, which matches the Type C tag, regardless of its serial number. In the second case, the pharmacist subscribes to an ECSpec considering only count view of tags grouped by their manager code. The subscription considers the current *DELETIONS* state of *shelf1*.
2: The ALE subscribes the Client to the *ReportsGenerator* that exclusively corresponds to the specified ECSpec (either for the nurse or the pharmacist applications).

```
<?xml version="1.0" encoding="UTF-8"?>
<ale:ECSpec xmlns:ale="urn:epcglobal:ale:xsd:1" >
<logicalReaders>
        <logicalReader>shelf1</logicalReader>
</logicalReaders>
<boundarySpec>
<repeatPeriod unit="MS">48300000</repeatPeriod>
<duration unit="MS">3000</duration>
</boundarySpec>
<reportSpecs>
        <reportSpec reportName="report-cardio-nurse">
                <reportSet set="DELETION"/>
                <filterSpec>
                        <extension>
                                <filterList>
                                        <filter>
                                                <includeExclude>INCLUDE</includeExclude>
                                                <fieldspec>
                                                        <fieldname>epc</fieldname>
                                                </fieldspec>
                                                <patList>
                                                     <pat>urn:epc:pat:sgtin-96:3.0000389.000162.*</pat>
                                                </patList>
                                        </filter>
                                </filterList>
                        </extension>
                </filterSpec>
                <output includeTag="true">
                        <extension>
                                <fieldList>
                                        <field>
                                                <fieldspec>
                                                        <fieldname>age</fieldname>
                                                </fieldspec>
                                        <\field>
                                <fieldList>
                        </extension>
                </output>
        </reportSpec>
</reportSpecs>
</ale:ECSpec>
```

Figure 7.5 – ECSpec filtering to obtain Type C tags (nurse application)

3.1: The *ReportsGenerator* instantiates the *PrivacyController* to verify the content of the request depending on the privacy policy.
3.2: In the case, the purpose and the logical reader of the new ECSpec are authorized, an update process is handled to verify if the remaining fields of the ECSpec corrrespond to the privacy policy.
4.1/4.1.1: The *PrivacyController* checks the Client request. When the request does not meet the privacy policy (e.g., more than the Type C medication is specified for that URI Recipient), or if the updated ECSpec is not correct, a deny message is returned to the ALE interface stating that the privacy policy does not allow the present subscription.
4.2: The *PrivacyController* checks the request and possibly update it. In the present case, the access is granted, and therefore, the *PrivacyController* allows the *ReportsGenerator* related to that event cycle to continue the processing.

```
<?xml version="1.0" encoding="UTF-8"?>
<ale:ECSpec xmlns:ale="urn:epcglobal:ale:xsd:1">
<logicalReaders>
        <logicalReader>shelf1</logicalReader>
</logicalReaders>
<boundarySpec>
<repeatPeriod unit="MS">302400000</repeatPeriod>
<duration unit="MS">3000</duration>
</boundarySpec>
<reportSpecs>
        <reportSpec reportName="report-pharmacist">
                <reportSet set="CURRENT"/>
                <groupSpec>
                        <pattern>urn:epc:pat:sgtin96:3.X.X.*.*</pattern>
                </groupSpec>
                <output includeCount="true"/>
        </reportSpec>
</reportSpecs>
</ale:ECSpec>
```

Figure 7.6 – ECSpec filtering to obtain tags numbers (pharmacist application)

5: The *ReportsGenerator* starts the *EventCycle*. The *EventCycle* now processes tags from the readers.
6/7: The LogicalReader(s) add(s) Tags to the *EventCycle*.
8/9: When the *EventCycle* reaches its boundaries, the reports are generated and sent back to the Client through the *ReportsGenerator*. The results are given in the following Section.

7.2.3 Obtained Results

Figures 7.7 and 7.8, respectively show the obtained ECReports, formatted in XML, which are related to the nurse and the pharmacy applications. Regarding the nurse application, results show the set of tags that are read by the *shelf1* reader. In the nurse specification, cf. Figure 7.5, the nurse searched to collect either the EPC code *epc* in the `filedname` attribute and a field named *age* in the memory content of the tag. This latter field has not been taken into account in the obtained report, as it is not specified in the data owner preferences. The

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ale:ECReports totalMilliseconds="3000" terminationCondition="DURATION" specName="spec-nurse" date="2013-04-
03T17:31:27.913+02:00" ALEID="ETHZ-ALE-750360204"  xmlns:ale="urn:epcglobal:ale:xsd:1" >
        <reports>
                <report reportName="report-cardio-nurse">
                    <group>
                        <groupList>
                                <member><tag>urn:epc:pat:sgtin-96:3.0000389.000162.1000</tag></member>
                                <member><tag>urn:epc:pat:sgtin-96:3.0000389.000162.1001</tag></member>
                        </groupList>
                    </group>
                </report>
        </reports>
</ale:ECReports>
```

Figure 7.7 – ECReports for the cardiology application

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ale:ECReports totalMilliseconds="3000" terminationCondition="DURATION" specName="spec-pharmacist"
date="2013-04-03T18:31:01.913+02:00" ALEID="ETHZ-ALE-750360204" xmlns:ale="urn:epcglobal:ale:xsd:1" >
<ale:ECReports xmlns:ale="urn:epcglobal:ale:xsd:1" >
        <reports>
                <report reportName="report-pharmacist">
                    <group name="urn:epc:pat:sgtin-96:3.0000389.*.*">
                        <groupCount><count>50</count></groupCount>
                    </group>
                    <group name="urn:epc:pat:sgtin-96:3.0000456.*.*">
                        <groupCount><count>4</count></groupCount>
                    </group>
                    <group>
                        <groupCount><count>60</count></groupCount>
                    </group>
                </report>
        </reports>
</ale:ECReports>
```

Figure 7.8 – ECReports for the pharmacy application

report shows that only EPC tags of type C are collected and presented with their entire code. The `DELETION` set specifies that reports must include only the tags that are taken away from the shelf, compared to the last event cycle.

Concerning the pharmacy application, the outputs show only the quantity of tags in the area of *shelf1*. This quantity is partionned in a number of groups differentiated by the manager code. The latter group shows the total tags that are in the frame of the filter specified in the pharmacist ECSpec (cf. Figure 7.6). The `CURRENT` set of tags specified in the ECSpec show the tags presented in the *shelf1*. In this case, the report is not updated as it is equivalent with the preferences specified by the data owner.

7.3 Performance Evaluation

We evaluate the performance of the proposed privacy enhanced middleware in terms of execution time. The aim of our simulations is to measure the total time needed for the middleware to serve all the requests. For this, we vary the number of simultaneous subscriptions to the middleware from 1 to 50, then, we consider the maximum response time among all the launched requests. We also consider in this test that the number of patients managed by the PrivOrBAC server is equal to 100. We compare our results with the original Fosstrak middleware, i.e., without modifying the *ReportsGenerator* entity in the server and the corresponding Capturing application. Note that our simulations are performed on a PC with Ubuntu OS 64 bits, 2.4 GHz of CPU and 4.0 GB of RAM.

Two scenarios are considered: the first one, named Verif. scenario, performs only a request verification to check the conformity of the declared purpose and the ECSpec fields with the data owner preferences (with a call to the PrivOrBAC entity). In this scenario, we assume that there is no need to update the ECSpec request. The second scenario, named Verif. &

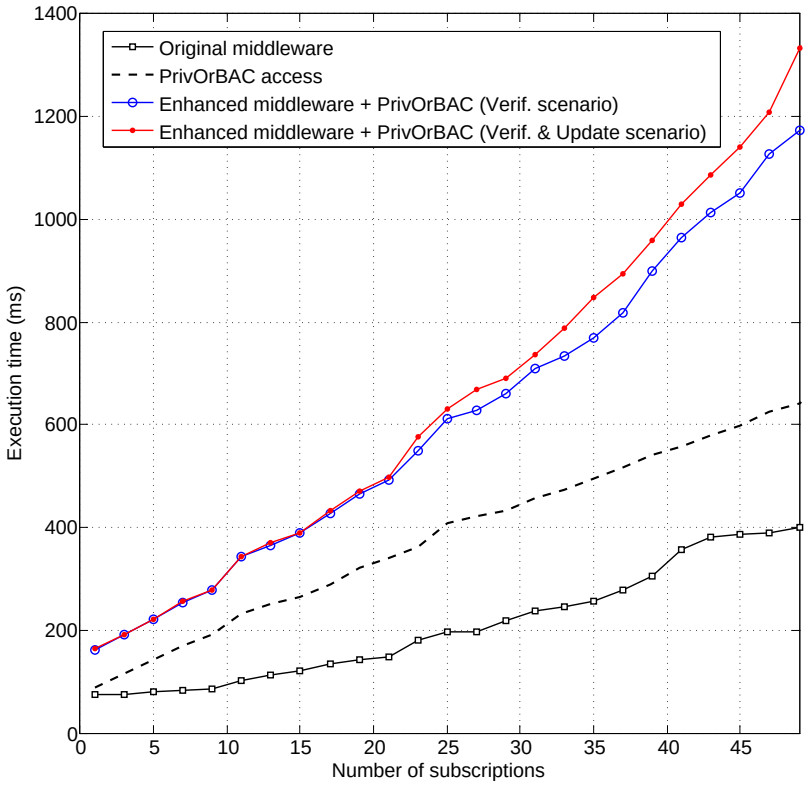


Figure 7.9 – Maximum execution time comparison with PrivOrBAC access (in Msec.)

Update, performs both the verification and the update of the ECSpec request according to the privacy policy.

Figure 7.9 compares the total execution time of the three studied cases (i.e., original scenario, meaning without Privacy control, Verif. scenario, and Verif. & Update scenario). As expected, we can notice that the total execution time needed in the privacy enhanced middleware to handle all the requests is higher than the original case, with relevant differences when the number of simultaneous subscriptions (N_s, for short) is high. This increase is mainly due to the execution time when accessing the PrivOrBAC server, which grows linearly with N_s (cf. Figure 7.9). In addition, we can observe that the execution time in the Verif. & Update scenario almost coincides with the execution time of the Verif. scenario when N_s is low (i.e., $N_s < 23$). The difference between the two scenarios becomes noticeable only when N_s is high. This means that the update time of the request in our privacy enhanced middleware is prominent only at high load (i.e., when the number of simultaneous subscription and ECSpec updates becomes higher than 33).

Figure 7.10 further investigates the execution time of the three studied cases (i.e., original scenario, Verif. scenario, and Verif. & Update scenario) in the middleware without accounting for the time needed to access the PrivOrBAC server. We can see that the execution time increases with N_s for the three cases. In particular, for both privacy enhanced scenarios,

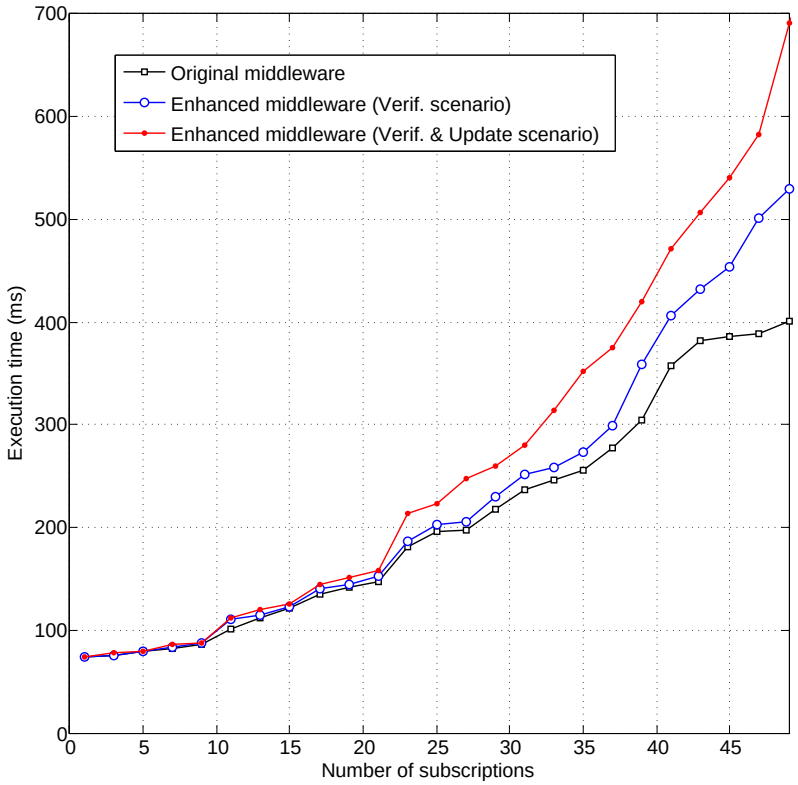


Figure 7.10 – Maximum execution time comparison without PrivOrBAC access (in Msec.)

the execution time slightly increases compared to the original case when the number of simultaneous subscriptions is low (i.e., $N_s < 23$). However, the gap becomes significant as N_s increases. Indeed, in the Verif. scenario (respectively, the Verif. & Update scenario), the execution time becomes clearly higher than the time for the original case when $N_s \geq 39$ (respectively, $N_s \geq 23$).

It is worth noting that the reported time measurements can be reduced when using a dedicated server with high computation capabilities, compared to the computer characteristics used for these simulations.

Conclusion

We implemented a privacy extension into the filtering and collection middleware that we added to the Fosstrak platform. Fosstrak is an RFID middleware implementation supporting the EPCglobal standard. After presenting the Fosstrak architecture and its main classes, we deduced that *ReportsGenerator* is the key point where event cycle specifications are launched after their verification. In addition, at this point, related reports are generated and sent to the subscribers. We presented the modifications performed at this level and exposed a part

of the Fosstrak code to handle the subscription action. To prove its applicability, we tested our prototypical solution and provide some performance evaluations in terms of maximum execution time of subscriptions. We considered two scenarios: the first one performs only a request verification to check the entered ECSpec fields, calling the PrivOrBAC server. The second scenario performs both the verification and the update of the request according to the privacy policy. Results showed that for both scenarios, the execution time in our privacy enhanced middleware slightly increases compared to the original middleware case when the number of simultaneous subscriptions is low. However, the gap becomes significant as the number of simultaneous subscriptions increases. In addition, compared to the first scenario, we observed that the update time of the request in the second scenario is prominent only at high load (i.e., when the number of simultaneous subscriptions and ECSpec updates becomes higher than 35).

CHAPTER 8

Conclusion and Perspectives

"It is the duty of every man to render to the world at least as much as he received, " – Albert Einstein

The target of most healthcare applications is to improve the quality of their systems to ensure substantial cost saving and to guarantee the minimum of privacy. Wireless technologies play a fundamental role in assisting people with various diseases in their everyday life. They help providing patients some quality of care, while keeping their quality of life. For example, they can be applied to ensure a reminder system for elders [82], or to permanently assist patients suffering from chronic diseases, who must travel regularly for minor typical examinations [94]. Radio Frequency IDentification (RFID) is an example of those technology. RFID allows contact-less identification of tagged objects and people. This low-cost technology is based on battery-less devices. It presents several advantages such as cost savings, small volume, fast data rate to access and collect information, besides to an easy attachment to the personal belongings. These advantages are considered attracting to help remote monitoring persons in need, by improving both patient safety and medical staff processing [109].

Although the RFID technology offers several benefits, its deployment still presents a variety of security and privacy implications. These implications are of importance when related to personal information of patients. In this sensitive context, the first objective of this thesis is to maintain secure communication between RFID readers and EPC Gen2 tags. By secure communication, we mean security against data disclosure threats in the presence of a strong adversary. This reader and tag communication is of real issue in low-cost RFID systems, which have to remain cheap while providing secure protocols that fit to the limited resources of the tag. A particular emphasis is made on the EPC Gen2 system which security model for accessing tag data is vulnerable to basic attack models. We have shown that the Gen2 model takes in account the different properties of the reader-to-tag channel and the tag-to-reader channel to distribute the security keys. This assumption does not suffice to bypass eavesdropping attacks, permitting a capable adversary, e.g., readers with high sensitive receivers and multiple antennas, to obtain the secret key sent by the tag. We proposed, for this aim, a security protocol that handles the problem of tag memory access to prevent data leakage.

The second objective of this thesis is to ensure that the data collected from the readers and sent to the back-end applications remain secure against rogue adversaries, typically curious applications aiming to disclose the patients personal data. We have shown that the middleware level is the key point for both data collection and request configuration. Despite its central location between readers and final database applications, the middleware level has not received much attention so far, in most of the middleware implementations available today. This leaded us to modify the existing EPCglobal middleware standard to satisfy the basic principles of privacy.

8.1 Main Results

The main results of this dissertation are stated as follows:

- A new key establishment and derivation protocol for Gen2 systems is designed. Its goal is to share and refresh keys in readers and tags to protect the privileged access to the tag memory. The tag memory may store secret keys and other sensitive information (e.g., related to the bearer of the tag). We proposed an alternative model to the EPC Gen2, assuming the use of a pseudorandom number generator (PRNG) whose algorithm and internal states are known at both reader and tag sides.

- An integration of the hand cipher Solitaire as a pseudo-random number generator (PRNG) in the KEDGEN2 protocol, is proposed. To adapt it to KEDGEN2, the cycle length (named orbit size in the group theory) of the Solitaire keystream has been estimated with respect to the size of the initial state. This cycle length is used as a threshold to update the internal state in the protocol. We have found that, differently from classical generators, the cycle length in Solitaire is variable depending on the cards positions (the internal state). The adaptation was performed by modifying the Solitaire algorithm depending on the maximum cycle length.

- A formal verification of the KEDGEN2 protocol shows that the current version of the protocol successfully handles the properties of mutual authentication and forward secrecy under a strong adversary model. The backward secrecy is also guaranteed under a modified but realistic adversary model that captures the capabilities of a real world adversary in typical RFID environments. We used the Constraint-Logic based Attack Searcher (CL-AtSe) as a model checker tool to specify the protocol, the adversary and the security properties.

- A privacy controller module that enhances the filtering and collection middleware of the RFID EPCglobal network is proposed. The privacy model is policy-driven using some enhanced contextual concepts of the extended Role Based Access Control model. It ensures the following features: (i) enforcing a privacy policy without interfering with

the standard middleware interface, (ii) using an existing privacy-aware model to store and manage privacy policy preferences, and (iii) taking into account the principles of purpose, consent, and collection limitation (accuracy) as privacy requirements.

- A proof-of-concept integrated into the middleware of the Fosstrak framework is provided. Fosstrak [69] is an open-source implementation of the EPCglobal specifications, which is recommended by the EPCglobal Inc [65]. Results of our simulations showed that the execution time in our privacy enhanced middleware slightly increases compared to the original middleware case when the number of simultaneous subscriptions is low. The gap becomes prominent only at high load (i.e., when the number of simultaneous subscriptions and ECSpec updates is high).

8.2 Perspectives

We give a set of future research directions that could be investigated as continuation of the results presented in this thesis.

Verifying the security properties with different model checking tools: The verification of our protocol was done using the Constraint-Logic based Attack Searcher (CL-AtSe) as a model checker. This latter takes in account the algebraic properties of the XOR operator. It adds positive and negative constraints on the intruder knowledge, variable values and sets, to see whether some security properties has been violated, by only limiting the number of sessions. As future work, we plan to use other verification tools such as ProVerif. This tool uses other checking methods (e.g., based on tree automata and Horn clauses) to search for security flaws in the protocol.

Finding the optimal threshold for state rekeying: We have observed that the update of the deck state before the maximum length cycle is reached, slightly enhances the statistical results of the Solitaire original outputs. But still, the optimum period of rekeying the Solitaire deck should be found, in order to provide better output statistics, compared to a true random number generator, for long time secure keys.

Performing an initial key establishment from a random source: In our model, we assumed that the initial state of the internal PRNG are generated from the reader side and with good statistical properties. The states updates, in our tests, were conducted using a C library that distributes initial states in a pseudo-random looking way. This clearly impacts the randomness quality of Solitaire outputs. A possible future direction is to use a true source of randomness that could renew the initial state in Solitaire algorithm in an unpredictable way. Then, to observe the statistical quality of the outputs.

Conducting hardware simulations: To evaluate the randomness of Solitaire cipher outputs, a statistical tests were conducted. The hardware implementation is also within our future research. Conducting SPICE simulations to extract power-consumption characteristics of the solitaire PRNG are an example of such evaluation. The aim is to check the suitability of our design for constrained environment of passive RFID tags. This allows us to bring some insight on optimizing the proposed design.

Introducing a robust updating algorithm: In the middleware level, we used a basic updating algorithm to rewrite ECSpec queries. It compares ECSpec fields with the preferences entries of the data owner. We rewrote the ECSpec to assure an ECReport, with maximum information with respect to the request. As a continuity of our approach, we will establish a rewriting algorithm that ensures all the correctness criteria. Current rewriting algorithms for fine-grained access control use the correctness criteria cited in [181]. The author, basically proposed their notions for relational databases. Three correctness criteria should be satisfied by any query processing algorithm: (i) the *Soundness* criteria, when the rewritten ECSpec returns only correct answers, i.e., answers to the initial query. (ii) the *Maximality* criteria, when the rewritten ECSpec returns as much information as possible and, (iii) the *Security* criteria, if the result of ECSpec respects the security and privacy policy of the queried system. Thus, as future work, we intend to find a trade off between the different correctness criteria related to the updating algorithm, as in our approach, the priority is given to the maximality criteria.

Treating the untraceability property: Untraceability assures that an attacker cannot trace the movement of a tag. This issue is observed when the person moves to different places. This issue does not present a real concern in environments of home healthcare systems, as the patients movement is limited to a small area. However in other contexts, this could be of real concern. This problem can be treated by introducing additional policy rules when configuring the logical readers. In the middeware level (i.e., specifically, using the Reading API), these logical readers are just called (i.e., further configurations of the readers is done with the middleware logical reader API, defined by EPCglobal). The establishment of untraceability rules for privacy reasons is another future research direction that could be investigated in the middleware level.

Aggregating several middleware events: When several middleware events go to one aggregation entity, a privacy-preserving problem could be raised, even if the privacy is handled separately in each middleware. This is due to the different security rules followed by each middleware. As a future work, we aim to find a trade off between data privacy preservation and secure data aggregation, in such centralized architectures.

APPENDIX A

The Block Cipher, a Pseudo-random Permutation

Security of block ciphers

The security provided by a block cipher depends on the security against key recovery and the security of the pseudo-random permutation E. It should be hard to distinguish the input/output behaviour of E_K from a random function without knowing the key K. We fix a block cipher $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ with key-size k and block size n.

Security against key recovery

Classically, the security of block ciphers has been related to key recovery. That is, the analysis of a block cipher E is done by considering some number q of inputs and outputs examples $(M_1, C_1), \ldots, (M_q, C_q)$, and trying to find K. K is a random, unknown master key and $C_i = E_K(M_i)$ for $i = 1, \ldots, q$ and $M_1, \ldots, M_q$ are all distinct n-bit strings. The question is how hard is it for an attacker to find a master key K? Some typical attack strategies are considered in this scheme are named Known-Plaintext Attack KPA and Chosen-Plaintext Attack CPA. In the first attack, $M_1, \ldots, M_q$ are distinct, arbitrary and are not controlled by the adversary algorithm. However by analyzing a given block of plaintext and corresponding ciphertext, the attacker tries to extract useful information for the recovery of plaintext encrypted in different ciphertexts or secret keys. In the second, $M_1, \ldots, M_q$ are adaptively picked by the adversary algorithm. Given its ability to choose plaintexts and generating corresponding ciphertexts, the adversary algorithm accesses an *oracle* of the function E_K and feeds the oracle M_1, then gets back $C1 = E_K(M_1)$. The latter value lets the adversary adaptively decides on the value M_2. Thus, it feeds the *oracle* to get back $C2$, and so on. Even if CPA gives the adversary more power, these latter are not always realistic in practice. The generic and most used attack strategy that works against any block cipher is named exhaustive key search EKS or a brute-force attack. This attack always returns the corresponding key with the above

sample of input-output. In the worst case, an adversary uses 2^k computations of the block cipher to obtain the key. We can conclude that there is no block cipher which is perfectly secure. It is viewed as secure if the best key recovery attack is computationally infeasible, which means that it requires a value of queries q or a running time too large to make the attack practical. Thus, to make key recovery by EKS computationally infeasible, one must enlarge the key length k of the block cipher.

Exhaustive key search attack (EKS)

The adversary algorithm tries all possible keys $K' \in \{0,1\}^k$ until it finds the one that explains the input-output pairs. Let $K_1, \ldots, K_2^k$ be a list of all K-bits keys. Let $K \xrightarrow{\$} \{0,1\}^k$ be the searched key and let (M_1, C_1) an example that satisfies $E_K(M_1) = C_1$.

Exhaustive Key search based on one sample
algorithm $EKS_E(M_1, C_1)$
for $i = 1, ...2^k$ ***do***
if $E(K_i, M_1) = C_1$ ***then return*** K_i

The likelihood of the attack returning the searched key can be increased by testing more samples of input-output. Moreover, the computations can be performed in parallel.

Security against key-recovery is necessary but it has been proved that it is not sufficient for block ciphers as it can still be possible for an attacker to find a relation between the input and the output after some time of executions without knowing K.

Security of the pseudorandom permutation

The security of a block cipher is defined by the advantage an adversary has in distinguishing a real function E_k from a random permutation. Using even a very *good* block cipher, the encryption (e.g., under the common mode of operation CBC (Cipher Block Chaining)) becomes insecure once $2^{n/2}$ blocks are encrypted under the same master key [22]. At this point, partial information about the message begins to leak. This leads to what is named birthday attacks [21]. For example direct use of 64-bit block cipher usually enables to safely encrypt no more than 2^{32} blocks. The cost of such an attack depends only on the block length. As a solution, we can enlarge the block length n, so that the $2^{n/2}$ time is unpractical. However, this is not a practical solution for environment supporting devices with limited memory capacities.

Relevant improvement schemes

To overcome the limitation of birthday attacks and securely encrypt more than $2^{n/2}$ messages in block ciphers, two major improvements are shown in the literature: the master-key re-keying and the data-dependent re-keying.

The master-key re-keying. Authors in [6] propose a scheme that protects against birthday attacks by changing the master key before the threshold number of encryptions permitting the attack is reached. The results show that re-keying the master key every $2^{n/3}$ encryptions, increases the threshold to $2^{2n/3}$ encryptions. This means that with the re-keying scheme, one can safely encrypt more data. This re-keying scheme also minimises the amount of damage that might be caused by key exposure. The exposure of the current key could determine all future keys (if the adversary has followed all the transactions), but if well used, the system cannot reveal past master keys that have to remain computationally infeasible to predict for the adversary even given the current master key and state.

The data-dependent re-keying. To minimise the advantage of an adversary to recognise encrypted data with some generated keys particularly when $q \geqslant 2^{n/2}$, [22] show that E must not be a family of permutation. The idea is to turn a pseudo random permutation (e.g., a block cipher) to a pseudorandom function which has not to be invertible. The basic case of the proposition apply when the key k and the input x of the block cipher have the same length $k = n$. The changed function F is then defined as $F(k, x) = E(E(k, x), x)$, where E is the permutation function. Thus, $F_k(x) = E_{k'}$, where $k' = E_k(x)$. This change is called *data-dependent re-keying*. F is twice the cost of computing E since there are two applications of E for each application of F. Protocols that are not worried about computing cost can use this scheme. A general construction of the solution is found in [22].

APPENDIX B

EPC Data Representation

The different representations of the EPC specified in the EPC Tag Data Standard [75] are intended for use at different levels within the EPCglobal architecture framework.

- Pure Identity EPC URI. The primary representation of an Electronic Product Code is as an Internet Uniform Resource Identifier (URI) called the Pure Identity EPC URI. The Pure Identity EPC URI is the preferred way to denote a specific physical object within business applications. The pure identity URI may also be used at the data capture level when the EPC is to be read from an RFID tag or other data carrier, in a situation where the additional control of information present in an RFID tag is not needed.
- EPC Tag URI. The EPC memory bank of a Gen 2 RFID Tag contains the EPC plus additional control information that is used to guide the process of data capture from RFID tags. The EPC Tag URI is a URI string that denotes a specific EPC together with specific settings for the control information found in the EPC memory bank. In other words, the EPC Tag URI is a text equivalent of the entire EPC memory bank contents. The EPC Tag URI is typically used at the data capture level when reading from an RFID tag in a situation where the control information is of interest to the capturing application. It is also used when writing the EPC memory bank of an RFID tag, in order to fully specify the contents to be written.
- Binary Encoding. The EPC memory bank of a Gen 2 RFID Tag actually contains a compressed encoding of the EPC and additional control information in a compact binary form. There is a 1-to-1 translation between EPC Tag URIs and the binary contents of a Gen 2 RFID Tag. Normally, the binary encoding is only encountered at a very low level of software or hardware, and is translated to the EPC Tag URI or Pure Identity EPC URI form before being presented to application logic.

Note that the Pure Identity EPC URI form is independent of RFID, while the EPC Tag URI and the Binary Encoding are specific to Gen 2 RFID Tags because they include RFID-specific control information in addition to the unique EPC identifier.

The figure below illustrates where these forms normally occur in relation to the layers of the EPCglobal Architecture Framework.

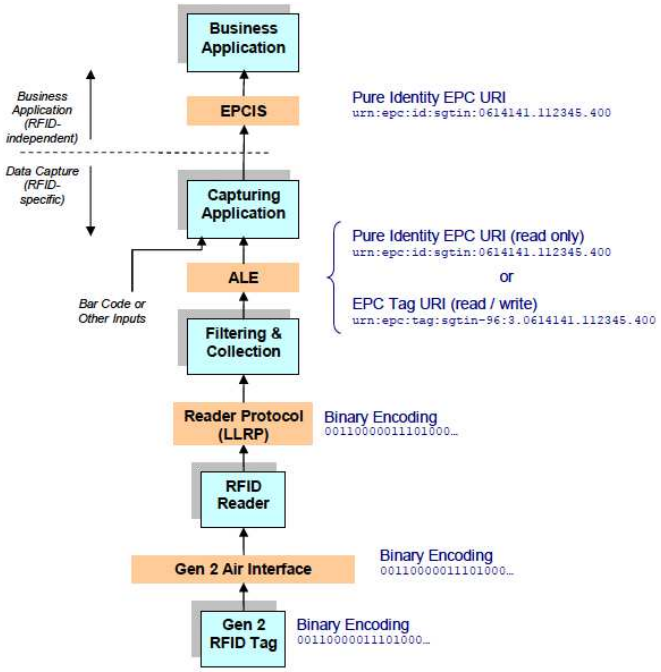


Figure B.1 – EPC-based data representation (taken from [11])

APPENDIX C

The Correctness Criteria

Wang et al. [181] proposed a formal notion of correctness for fine-grained access control in relational databases. They presented three correctness criteria that should be satisfied by any query processing algorithm in order to be "correct".

Soundness : An algorithm is sound if and only if its rewritten query Q_{rw} returns only correct answers i.e. answers to the initial query.

Maximality : An algorithm is maximum if and only if Q_{rw} returns as much information as possible.

Security : An algorithm is secure if and only if the result of Q_{rw} respects the security and privacy policy of the queried system.

In the rest of this section, we present a condition that should be satisfied by our algorithm in order to satisfy the soundness, maximality and security criteria [181].

Let A be a query processing algorithm that enforces privacy policies. Let D be a database, P a disclosure policy and Q a query. Let $Q' = A(P, Q)$ be the rewriting query of Q using the policy P. We denote $R = A(D, P, Q)$ the output result of Q' on D.

Definition 1 : Given two tuples $t_1 = (x_1, x_2, ..., x_n)$ *and* $t_2 = (y_1, y_2, ..., y_n)$*, we say that* t_1 *is subsumed by* t_2*, denoted* $t_1 \sqsubseteq t_2$*, if and only if* $\forall i \in [1..n] : (x_i = y_i) \vee (x_i = unauthorized)$.

Definition 2 [181] : Given two relations R_1 *and* R_2*, we say that* R_1 *is subsumed by* R_2*, denoted* $R_1 \sqsubseteq R_2$*, if and only if:*

$$(\forall t_1 \in R_1)(\exists t_2 \in R_2)|t_1 \sqsubseteq t_2$$

Definition 3 [181] : Two databases states D *and* D' *are "equivalent" with respect to policy* P *(denoted as* $(D \equiv {}_P D')$ *if the information allowed by* P *in* D *is the same as that allowed by* P *in* D'.

Let S denote the standard query answering procedure and $S(D, Q)$ the result of the query Q in the database state D without any privacy restriction. [181] formalizes the three criteria as follows: a query processing algorithm A is:

- sound if and only if :$\forall P \forall Q \forall D \quad A(D, P, Q) \sqsubseteq S(D, Q)$
- secure if and only if:$\forall P \forall Q \forall D \forall D' \quad [(D \equiv {}_P D') \longrightarrow (A(D, P, Q) = A(D', P, Q))]$
- maximum if and only if: $\forall D' \forall R$
  if $[(D \equiv {}_P D') \wedge (R \sqsubseteq S(D', Q))]$ then we have $R \sqsubseteq A(P, D, Q)$

These correctness criteria should be used when updating the original request ECSpec with a new filtered one, in the middleware level.

APPENDIX D

Deploying the Fosstrak ALE Middleware with the LLRP RFID Reader

Prerequisites

Directory Trees of the fc-server and fc-common Directories

After changing the Fosstrak Filtering and Collection server (*fc-server*) and the Web-based client to support the *purpose* attribute as an entry along with the ECSpec, and the URI address, a compilation of the server is required. Figure D.1 shows a tree structure of the main changes between the original version of *fc-server* (at the left side) and the modified one (the right side). We have also changed the *fc-common* directory (Figure D.2) to integrate exception treatments (in relation with the right to collect data). Note that the files colored in red mean that they are modified whereas the files colored in green mean that they are added to the tree structure.

To perform a data collection, the following files are needed:

- a compiled fc-server-1.2.0.war (after the modification)
- a compiled fc-webclient fc-webclient-1.2.0.war (after the modification)
- a copy of the reader protocol client version 0.3.1
- a copy of the LLRP ADD_ROSPEC message [1]
- an ALE Event Cycle Specification message
- an ALE logical reader definition file

[1] http://fosstrak.googlecode.com/svn/wikires/ale/ROSPEC_example.llrp

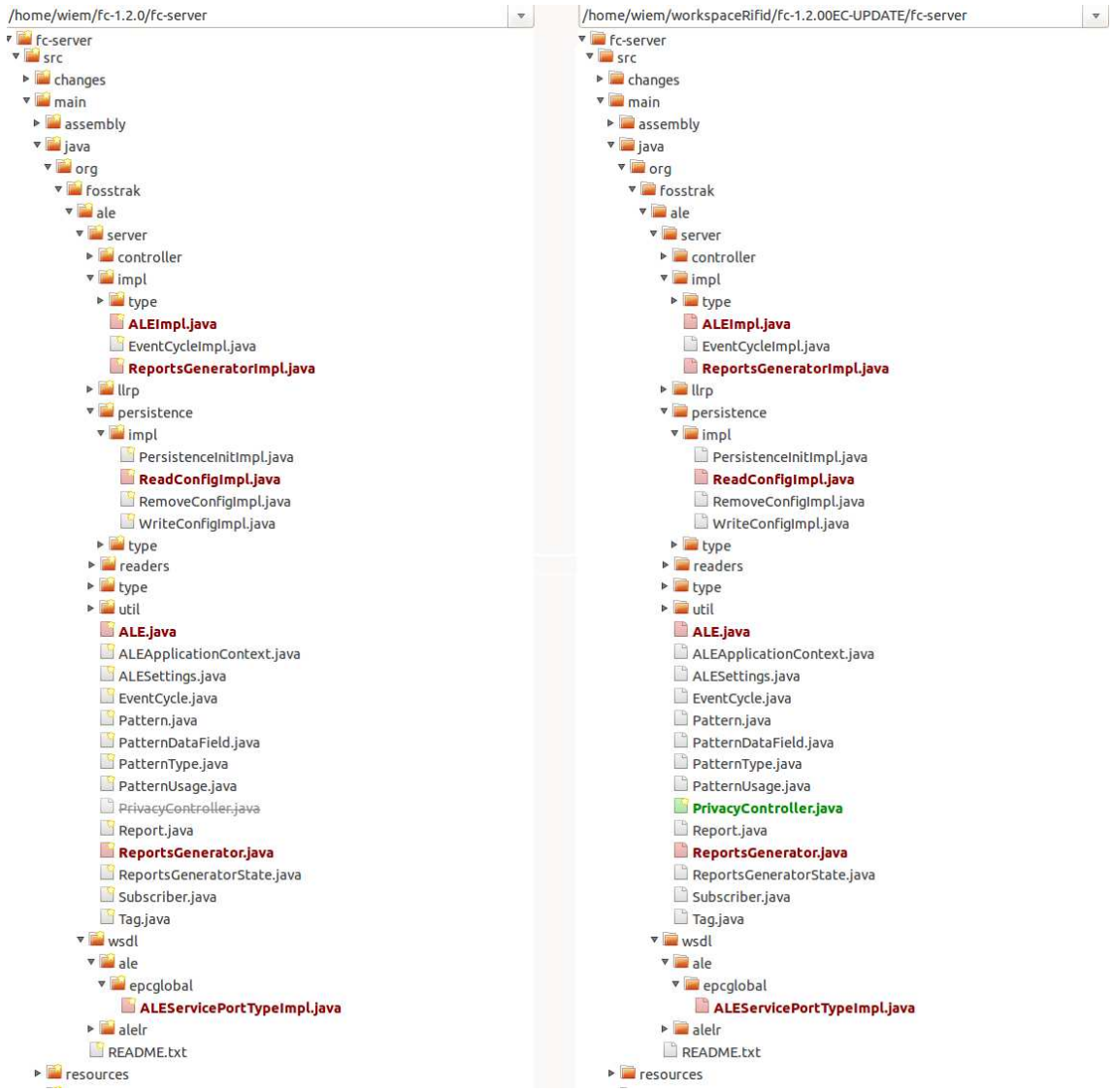


Figure D.1 – Tree structure of both the original and the modified *fc-server* version

The following tools should be initialized:

- Fosstrak LLRP Commander
- Rifidi Reader Emulator
- Apache Tomcat Java Servlet Container

In the sequel, we give the details of the steps to be followed.

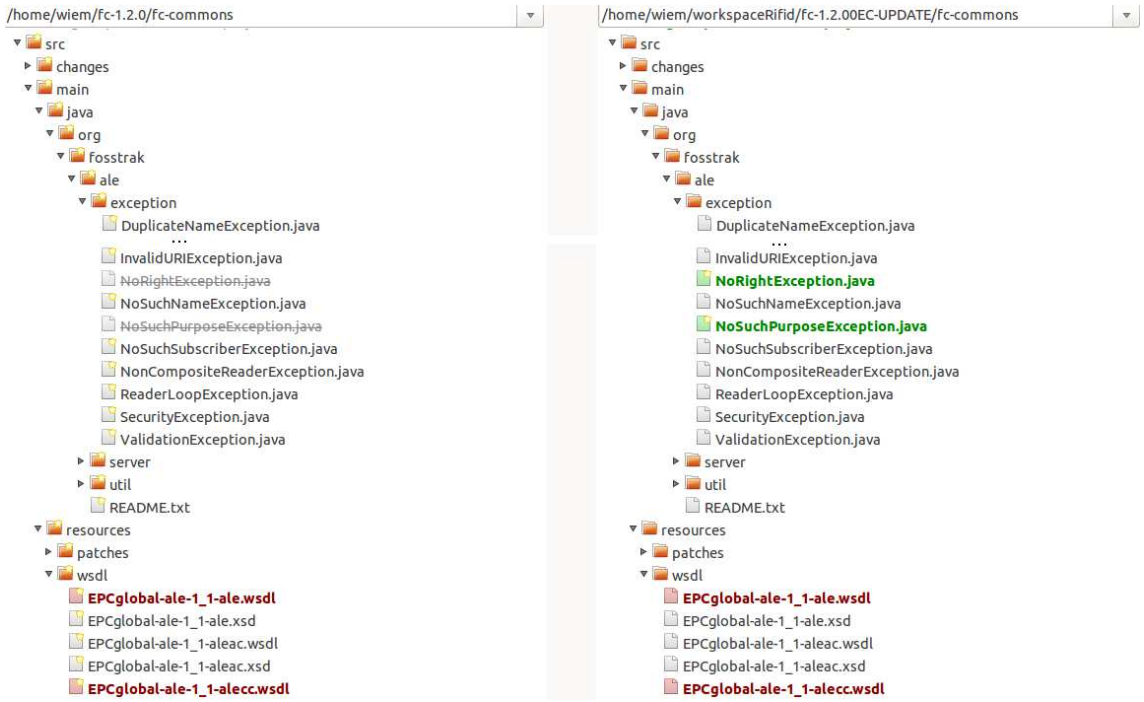


Figure D.2 – Tree structure of both the original and the modified *fc-common* version

Testing the Modified Middleware

Prepare Tomcat

After installing Apache Tomcat, the compiled fc-server-1.2.0.war and the fc-webclient-1.2.0.war should be copied into the Tomcat's webapps directory. After starting Tomcat, it will automatically deploy the two WAR files.

Setup LLRP reader Emulator

After installing and starting the Rifidi Emulator, instantiate a new reader of type *LLRPReader* that listen on port 5084 (the default port) using the Reader Wizard. Then, start the reader just instantiated.

When configuring the other components, we will return to the LLRP reader Emulator.

Setup GUI for incoming HTTP ALE notifications

To display the reports sent by the F&C middleware server, start a Fosstrak tool to display incoming HTTP requests.

```
java -cp <READER_RP_CLIENT_VERSION>.jar \
```

```
org.fosstrak.reader.rp.client.EventSinkUI <PORT>
```

Configure the Fosstrak ALE web-client

The web-based client of Fosstrak is used here. We specify the URL through which the version of the compiled F&C server will be accessed. In our case, we specify:

```
http://localhost:8080/fc-webclient-1.2.0/services/ALEWebClient.jsp
```

Then, we specify two endpoints to point the location of the "F&C middleware API" and the "Logical Reader API".

The endpoint to the compiled F&C server (*fc-server*) is set by invoking the `setEndpoint(String endPointName)` method in the F&C API:

```
http://localhost:8080/fc-server-1.2.0/services/ALEService
```

The endpoint to the F&C server's Logical Reader API is set by invoking the method `setEndPoint(String endPointName)` in the LogicalReader API:

```
http://localhost:8080/fc-server-1.2.0/services/ALELRService
```

Set the connected readers to ALE Middleware via the ALE Logical Reader API

Next, the configuration of the *fc-server* with the LLRP reader connected, is required. The `define(String readerName, LRSpec spec)` method in the section LogicalReader API, is pointed. The name of the reader is for example: "LogicalReader1" using the LLRPReader.xml as LRSpec:

```
readerName: LogicalReader1
specFilePath: home\laptop\epc\LLRPReader.xml
```

After this step, the messages exchange with the Fosstrak ALE middleware should be observed in the Rifidi Emulator.

Define the F&C behavior via the ALE middleware API

Here a definition of an ALE ECSpec is done. This tells the ALE middleware how the RFID tag reads, from the Rifidi Emulator should be filtered and aggregated. The definition of the EC-Spec is done by invoking the method `define(String specName, String specFilePath)`. The name of the ECSpec is for example: "Spec_Nurse":

```
specName: specCURRENT
specFilePath: home\laptop\epc\ECSpec_current.xml
```

Subscribe to the ECSpec

In this step, we specify the listener (or the event consumer) of the ALE Events specified by the ECSpec. For this, the method `subscribe(String specName, String notificationUri)` have to be invoked after registering the URL on which the Fosstrak event sink GUI is listening. The ALE will start sending empty ECReports to the event sink GUI, if the Rifidi Emulator is not configured to send EPC tag reads via the LLRP protocol.

Configure the Rifidi Emulator to report tags in the reader range

The LLRP Commander is started via Eclipse, and connected to the remote adapter instance running on the ALE.

The first step is to create a new LLRP message and replace the content by the content from the LLRP RO_SPEC specified in the Prerequisites. Then, send the message to the Rifid Emulator. (from the context menu, select `Send ENABLE_RO_SPEC` message to instruct the reader to enable the ROspec just loaded.

Switch to the Rifidi reader Emulator and create a tag (e.g., SGTIN96, GEN2). When the tag is placed on the reader antenna, the tag reads will be reported to the Fosstrak ALE Middleware. After the tag reads are filtered and collected as specified in the ECSpec, they are delivered by the Fosstrak ALE middleware to the event sink GUI.

APPENDIX E

Résumé de la Thèse

Titre: Sécurité et Protection de la Vie Privée dans les Systèmes RFID, Appliquées aux Réseaux EPCglobal

Radio Frequency IDentification (RFID) s'est émergé comme une technologie à faible coût due à l'utilisation de dispositifs sans batteries. Elle est connue comme le successeur des omniprésents codes à barre. Contrairement aux codes à barre optiques, les dispositifs RFID permettent l'identification individuelle des objets, beaucoup plus rapidement, même à travers des obstacles et sans l'obligation d'une intervention humaine. Ces dispositifs sont nommés les étiquettes RFID passives. Elles peuvent être fixées ou intégrées à un objet à identifier et sont lues quand elles entrent dans le champ d'un lecteur RFID. EPC (Electronic Product Code) de classe 1 et Génération 2 (Gen2) est un exemple typique de la technologie RFID des étiquettes passives. Le coût relativement faible de cette technologie RFID passive, lui a permis d'être attractive pour les départements de logistiques qui cherchent des possibilités pour intégrer cette technologie à temps réel dans leurs processus métier. Par ailleurs, il a été noté que cette technologie améliore significativement la visibilité et la précision des opérations logistiques [130]. Malgré sa large utilisation et ses applications prospectives, la technologie RFID pose plusieurs problèmes de sécurité et de vie privée, qui pourraient nuire à son adoption globale.

Contexte et Motivation

La technologie RFID a initialement été créée pour automatiser la collection des données dans les chaînes d'approvisionnement. Pendant ces dernières années, des progrès substantiels ont été réalisés pour intégrer cette technologie dans des applications sensibles au contexte (dites, *context-aware applications*). L'utilisation des radio-étiquettes passives RFID pour implémenter des systèmes de suivi de la santé à distance est l'un des scénarios prometteurs pour ce type d'applications. En effet, l'utilisation de la technologie RFID dans un domaine aussi critique connaît un succès croissant avec le besoin d'assister les personnes (exemple les personnes agées et les malades) dans leurs vie quotidienne [82, 94]. D'après une étude réalisée

par le bureau de la censure aux États Unis en 2000 [137]: *le bilan net des personnes âgées dans le monde a augmenté de plus de 750,000 par mois; Dans deux décennies, l'augmentation serait probablement de deux millions par mois.* Aussi, il a été noté dans [28] que les patients âgés, souffrant de maladies chroniques sont les plus adhérents à des services de santé à domicile. Enfin, une découverte cruciale dans [60] citait que les patients insistent sur la nécessité de se sentir en contrôle, en plus de leurs désir d'être entourés d'une ambiance agréable. Toutes ces raisons montrent le besoin du patient d'être suivi à distance par des systèmes de contrôle efficaces. D'une autre part, les infirmières qui doivent régulièrement visiter leurs patients, peuvent compter sur ces nouveaux systèmes pour alléger leur charge de travail et réduire les tâches administratives conséquentes [109]. Par exemple, l'utilisation de la technologie RFID contribue à assurer que le bon médicament est donné au patient au bon moment et dans le bon dosage. En conséquence, de plus en plus de systèmes de suivi de la santé à distance font usage de nouvelles technologies telles que les étiquettes RFID de faible coût, pour améliorer leurs qualité de service et assurer une économie des coûts considérable. Mais encore, ces solutions de suivi de la santé à distance, doivent être soigneusement conçues pour garantir un minimum de respect de la vie privée.

Le déploiement de la technologie RFID avec des étiquettes passives est devenu plus important avec la standardisation du processus via la norme des étiquettes EPC (Electronic Product Code) Class 1 Generation 2 [61] (connue par, Gen2). Conçue par des membres de l'MIT (Massachusetts Institute of Technology) et développé par le consortium EPCglobal [65], la technologie EPC représente un composant clé d'une architecture nommée, EPCglobal network [61]. Le standard EPCglobal couvre l'ensemble de l'architecture RFID [11]. De la spécification de la structure de données de l'étiquette aux communications dans le réseau. Le rôle principal de cette architecture est l'identification automatique de l'objet en mouvement (identifié par l'étiquette RFID) et la lecture du contenu de sa mémoire, pour partager cette information à travers un processus métier.

Le réseau EPCglobal est un ensemble de normes techniques mondiales, qui consiste, d'une vision très abstraite, en trois niveaux basiques (Voir Figure E.1): le niveau radio liant le lecteur RFID et les étiquettes, le niveau intergiciel (ou *middleware*), traitant les configurations et les données en temps réel et le niveau service d'information (EPCIS). Ce dernier niveau traite les données filtrées et agrégées pour les traduire à des événements métiers connexes et les rendre disponibles pour les accès internes et externes. La persistance des données est, ainsi, fournie à ce niveau par un dépôt EPCIS qui maintien les événements historiques. Malgré sa large utilisation, cette technologie pose plusieurs problèmes de sécurité et de vie privée qui peuvent affecter son utilisation. Par exemple, ces étiquettes peuvent contenir plus d'information qu'un simple identifiant comme l'âge du porteur ou les données médicales le concernant. Il convient de signaler que la technologie RFID, depuis sa conception, n'assure pas la protection de la vie privée, puisque elle avait pour objectif initial de permettre l'identification rapide et automatisée de l'objet lié à l'étiquette.

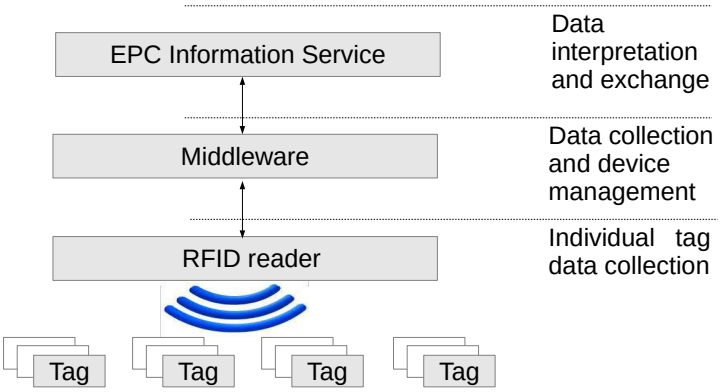


Figure E.1 – Les principaux niveaux dans l'architecture EPCglobal

Comme dans de nombreuses autres technologies émergentes, si des contre-mesures devant les attaques de fuites de données ou d'usurpation d'identité ne sont pas prises en compte au niveau le plus bas de l'architecture, les risques de sécurité et de violation de la vie privée augmentent. Dans cette thèse, nous nous focalisons sur l'utilisation des dispositifs RFID passifs, spécialement, les étiquettes Gen2. Nous traitons le problème des atteintes à la vie privée du porteur, spécifiquement les fuites de données, et ce, à travers deux niveaux de l'architecture de EPCglobal. Un niveau qui concerne l'échange de données dans le canal radio liant le lecteur RFID à l'étiquette passive et un niveau qui concerne le *middleware* pour la collection et le filtrage des données. La fuite des données implique la divulgation de l'identité de l'étiquette (c'est à dire EPC dans la technologie Gen2) et le contenu de sa mémoire additionnelle. Ces données soulèvent une réelle préoccupation, si elles sont liées à des informations personnelles, par exemple, la maladie du porteur de l'étiquette.

Dans ce contexte, de nombreux défis sont rencontrés. Les défis qui sont abordés dans cette thèse sont organisés en deux parties. La première partie concerne le protocole de communication reliant les lecteurs aux étiquettes dans l'infrastructure RFID (c'est à dire *the front-end*). La deuxième partie se concentre sur les problèmes de confidentialité soulevés dans la partie en aval de la même infrastructure (c'est à dire *the back-end*), en particulier, le composant d'intergiciel RFID, dit, *middleware*, pour le filtrage et la collecte des événements.

Ce chapitre est organisé comme suit. Dans la première partie, un nouveau protocole de communication entre le lecteur et les étiquettes est défini pour combler les limites du protocole proposé par le standard EPCglobal. Ensuite, un générateur de clés pseudo-aléatoire est proposé, en respectant les limites des étiquettes passives, en terme d'énergie et de capacité de calculs. Par la suite, la vérification formelle du protocole est donnée pour prouver son maintien de la forte confidentialité des données échangées entre le lecteur RFID et l'étiquette. En ce qui concerne la deuxième partie, l'intégration d'un module de préservation de la vie privée est proposé dans le *middleware* du standard EPCglobal. Une preuve de concept est montrée à travers l'implémentation du module dans la plateforme Fosstrak (une implémentation des spécifications du standard, recommandée par EPCglobal).

KEDGEN2: un protocole d'établissement de clé et de dérivation pour les étiquettes RFID passives

Les défis

Assurer la sécurité de la communication entre le lecteur RFID et l'étiquette est une tâche difficile. D'une part, la norme fournit une protection très basique pour les opérations privilégiées qui nécessitent une authentification du lecteur (par exemple, accéder à la mémoire de l'étiquette ou la désactiver). D'autre part, les capacités de l'étiquette RFID Gen2 sont très limitées pour intégrer à bord les solutions de cryptographie traditionnelle. En même temps, il est important de maintenir le faible coût des étiquettes RFID Gen2 pour promouvoir son utilisation à grande échelle.

Concernant le premier point, dans le dernier standard RFID, ratifié par EPCglobal, les étiquettes supportent un générateur de nombres pseudo-aléatoires (PRNG) et un contrôleur CRC (Cyclic Redundancy Check). Les mécanismes de protection de la confidentialité consistent principalement à rendre l'étiquette définitivement inutilisable une fois qu'elle reçoit la commande *kill* avec un mot de passe de 32 bits valide. L'accès privilégié comme la lecture/écriture dans la mémoire de l'étiquette est autorisé seulement après que l'étiquette ne rentre en mode sécurisé. Ce mode est accessible quand l'étiquette reçoit une commande d'accès avec un mot de passe de 32 bits valide (divisé en deux messages). À ce stade, une réelle menace est soulevée, en raison du modèle inefficace utilisé pour protéger ce dernier mot de passe d'accès. En fait, cette dernière transaction est réalisée par le lecteur RFID en appliquant une opération *OU exclusif* d'une clé pseudo-aléatoire (Voir Étape 3 dans la Figure E.2). Cette clé est déjà produite dans la même session et envoyée en clair par l'étiquette (Voir Étape 2 dans la Figure E.2).

1. Lecteur ⟶ Etiquette RFID : Demande de clé
2. Etiquette RFID ⟶ Lecteur : Clé secrète
3. Lecteur ⟶ Etiquette RFID : Mot de passe $\oplus$ Clé secrète

Figure E.2 – Demande d'accès privilégié à l'étiquette RFID

En utilisant ce protocole, il est simple qu'un adversaire capable d'écouter passivement l'échange, puisse extraire la clé pseudo-aléatoire envoyée par l'étiquette RFID à l'aide de dispositifs matériels particuliers (par exemple, les lecteurs avec des récepteurs de haute sensibilité et de multiples antennes). Il est, par ailleurs, simple pour cet adversaire capable de prédire la sortie du générateur de nombres aléatoires de l'étiquette (s'il est basé sur un générateur pseudo-aléatoire, statistiquement non robuste et compatible avec EPC Gen2 [120]) d'obtenir cette clé secrète. Ainsi, il peut récupérer le mot de passe d'accès en appliquant une seconde fois l'opération *OU exclusif*. L'obtention du mot de passe d'accès permet à l'adversaire malveillant d'accéder et de modifier la mémoire de l'étiquette. Lorsque la mémoire contient

des informations supplémentaires sur les objets qui peuvent être liées aux personnes qui les détiennent (par exemple, des informations relatives à un médicament particulier qui peut relier à une maladie particulière), la divulgation de l'information peut être une menace pour la vie privée. Cette menace peut être surpassée en modifiant le modèle de sécurité spécifié par Gen2 et en fournissant une manière plus élaborée pour la gestion des clés cryptographiques dans les systèmes Gen2. Par conséquent, le problème se révèle en rapport avec l'établissement de clés pour s'assurer de toujours utiliser une nouvelle clé de cryptage dans chaque nouvelle session. Cette clé sert de clé principale pour les dérivations suivantes (c-à-d. la génération de clé), au sein de la même session. Ce problème s'agit de l'objet de notre première contribution qui sera détaillée dans la section qui suit.

Concernant le deuxième point, malgré les efforts déployés pour adapter les méthodes de hachage ou de cryptographie à clé publique pour les étiquettes RFID passives, de tels algorithmes sont au-delà des capacités actuelles de ces étiquettes [26]. En conséquence, les protocoles cryptographiques pour les étiquettes RFID passives peuvent au mieux être construits en utilisant des primitives symétriques basées sur les fonctions de génération de clés. Ainsi, un nouveau défi de sécurité est lié à la fonction de génération de clé. La technologie RFID basée sur les PRNGs légers ne devrait pas générer des clés secrètes prévisibles (comme est le cas des premières propositions pour protéger les échanges faisant intervenir les étiquettes RFID [70, 160]). Notre objectif est alors d'utiliser un PRNG léger et de maximiser les chances de générer des clés, d'apparence aléatoire. Ce problème de sécurité s'avère aussi une protection de la vie privée, puisque l'étiquette RFID peut combiner les données sensibles avec une clé différente à chaque transaction.

La contribution

Un nouveau protocole d'établissement et de dérivation de clés, appelé KEDGEN2 a été défini pour les systèmes Gen2 de deuxième génération. Il est modélisé pour partager et rafraichir les clés entre le lecteur et l'étiquette RFID. Notre protocole se focalise sur l'accès privilégié à la mémoire de l'étiquette où les clés, ainsi que d'autres informations sensibles, pourraient être stockées. Nous proposons à travers ce protocole, l'utilisation d'un modèle alternatif à EPC Gen2, en proposant l'utilisation d'un générateur de nombres pseudo-alétoires, dont l'algorithme et les états internes sont connues par le lecteur et l'étiquette (cf. [171, 175]).

Puisqu' il a été démontré que les étiquettes RFID passives ont des capacités similaires aux êtres humains [91] et en raison de la faible complexité requise par le mélange des cartes dans le générateur de clé dans Solitaire [161], nous proposons d'adapter l'algorithme de chiffrement à la main Solitaire, comme un générateur de nombres pseudo-aléatoires (PRNG). Les clés générées sont utilisées dans le protocole KEDGEN2, comme des clés à usage unique (one time keys). Pour trouver le seuil de génération qui permet de mettre à jour l'état interne du générateur, nous cherchons la taille de cycle de Solitaire, nommée la taille de l'orbite dans la théorie des groupes. Nous avons constaté que différemment des générateurs classiques, la taille

de l'orbite dans Solitaire est variable en fonction des positions des cartes (l'état interne). Pour l'adapter à KEDGEN2, nous modifions Solitaire en mettant à jour l'état initial (la graine) par rapport à la taille maximale de l'orbite. Enfin, nous démontrons que la version modifiée de Solitaire peut être facilement intégrée dans notre protocole KEDGEN2 [176].

Vérification formelle du protocole KEDGEN2

Le défi

Même en utilisant un algorithme cryptographique fort dans un protocole de communication, cela ne garantit pas que le protocole sera sécurisé. En effet, une bonne cryptographie peut être mal utilisée en raison d'une mauvaise conception du protocole. De nombreux protocoles ont été proposés dans la littérature, cependant, la plupart d'entre eux s'est avérée, peu de temps après leurs publications, défectueuses. Les failles sont généralement liées au protocole. Le cas le plus connu est le WEP (Wired Equivalent Privacy), le premier protocole de sécurité pour les réseaux WiFi. Le chiffrement de flux RC4 utilisé dans WEP est remarquablement simple à mettre en œuvre et considéré comme *fort* s'il est utilisé d'une manière appropriée. Néanmoins, les concepteurs du WEP n'ont pas fait pareil. Comme de nombreuses analyses le concluent, les faiblesses du WEP ne proviennent pas des défauts de RC4, plutôt de la façon dont il est appliqué [59]. Ainsi, un besoin important de protocoles de sécurité prouvés et validés apparaît. Ces preuves peuvent être effectuées en suivant une approche d'analyse numérique ou symbolique. Ces preuves peuvent aussi être aidées par des outils automatiques, élaborés à cet effet. Malheureusement, ce n'était pas systématique dans la plupart des ouvrages traitant de la sécurité dans les RFID où les objectifs de sécurité ont été démontrés *intuitivement*. Cette vérification formelle du protocole représente un autre défi dans cette thèse, où l'authentification mutuelle de l'étiquette RFID et du lecteur et les propriétés *forward secrecy* et *backward secrecy* sont vérifiées. La propriété *forward secrecy* (respectivement, *backward secrecy*) garantit que l'exposition de la clé principale à l'adversaire, ne lui permet pas de calculer la précédente (respectivement, la future) clé principale, une fois cette dernière est actualisée.

La contribution

Pour vérifier la sécurité du protocole KEDGEN2 à l'égard des fortes notions d'authentification et de confidentialité, nous suivons une approche formelle. Plus précisément, nous utilisons un outil de *model checking*. *Model checking* est une technique de vérification formelle commune, basée sur une approche symbolique. Nous spécifions, d'abord, le protocole en utilisant le langage HLPSL (High Level Protocol Specification Language). Ensuite, nous vérifions les propriétés de sécurité, grâce à des techniques de *model checking* de l'outil CL-ATSE (Constraint-

Logic based Attack Searcher), développé dans les projets AVISPA/AVANTSSAR. Cet outil est connu pour sa maturité dans les techniques de résolution de contraintes [99].

La version actuelle du protocole KEDGEN2 garantit l'authentification mutuelle des participants et la propriété *forward secrecy* en présence d'adversaires actifs. Elle garantit aussi la propriété de *backward secrecy* en présence d'adversaires actifs délimités par des opérations de communication restreintes, compatibles avec les environnements typiques RFID [174].

Un modèle axé sur la politique de protection de la vie privée dans le *middleware* EPCglobal

La seconde partie de cette thèse concerne la protection des données contenues dans les étiquettes collectées par le composant d'intergiciel RFID (appelé, *middleware*). Le RFID *middleware* est un composant situé entre les lecteurs d'étiquettes et les bases de données applicatives. Il assure les fonctions de collecte, de filtrage et d'agrégation des événements remontées à partir des lecteurs d'environnements RFID hétérogènes. En conséquence, le système est susceptible à ce stade d'être manipulé au niveau du paramètrage des données à collecter ou d'être en écoute discrète, conduisant à des problèmes de confidentialité.

Le défi

Dans le processus de leurs collecte, les données contenues dans les étiquettes peuvent potentiellement être apprises par différentes applications clientes, par exemple des applications curieuses. Ceci permet vraisemblablement à des mécanismes de surveillance trés fins et même excessifs qui envahissent la vie des porteurs des étiquettes [44]. Par exemple, selon les directives du standard HIPAA [10] (Health Information Portability and Accountability), protéger les informations en rapport avec la vie privée du porteur des étiquettes signifie qu'un collecteur ne doit pas être en mesure d'identifier le type de produit en lisant simplement le code d'identification de l'étiquette RFID. Ainsi, l'identification pourrait être réalisée en tenant compte uniquement de certaines parties du code d'identification, comme le fabricant du produit ou la famille des étiquettes, en fonction de l'objectif du collecteur. La même question se pose lorsque certains champs dans la mémoire de l'étiquette sont lus. Si ce contrôle pourrait être réalisé avant que les événements générés ne soient envoyés à des applications de couches supérieures, où l'information est interprétée et stockée, il serait intéressant que le *middleware* RFID, qui est responsable de la configuration des lecteurs, considère cette préoccupation des consommateurs de la technologie RFID, ainsi que le cadre juridique de ce problème.

La plupart des implémentations de *middleware* disponibles aujourd'hui proposent un contrôle d'accès basé sur les rôles. Cependant, ils ne parviennent pas à résoudre les problèmes de confidentialité des consommateurs en soutenant de manière appropriée les directives internationales issues du FIP [67] (Fair Information Practices). Une raison possible de ce problème

est que la plupart des entreprises exécutent le *middleware* RFID dans un réseau interne [39]. En outre, comme expliqué dans [34], l'effort pour assurer la sécurité et la vie privée existe dans les travaux actuels sur le *middleware* RFID, cependant, ce n'est pas fait comme une incorporation architecturale, plutôt comme des règles de sécurité spécifiques à un domaine, mis en place au respect d'un *middleware* spécifique. Le dernier défi, dans cette thèse, est de montrer la possibilité de modifier le standard du *middleware* EPCglobal existant, afin de satisfaire les principes de base de la vie privée. L'objectif est de proposer une approche qui s'applique à toutes les applications où l'information sensible doit être conservée en toute sécurité. Par exemple, dans le contexte des applications de suivi de santé à domicile.

La contribution

Il est incontestatble que l'acte de lire une ou plusieurs étiquettes RFID constitue une collecte de données. Ainsi, les lois et règlements en vigueur sur la vie privée s'appliquent à la communication impliquant le *middleware* RFID.

Nous proposons une approche centrée sur l'utilisateur, qui peut définir la politique de vie privée s'appliquant aux données collectées et agrégées par le *middleware*. Elle intègre la mise en œuvre d'une politique de vie privée sans interférer avec l'interface standard du *middleware* (voir Figure E.3). L'expression des préférences de vie privée est faite en utilisant le modèle PrivOrBAC, en prenant en compte les dimensions d'objectif déclaré de la requête, de précision des résultats produits et de consentement explicite de l'utilisateur.

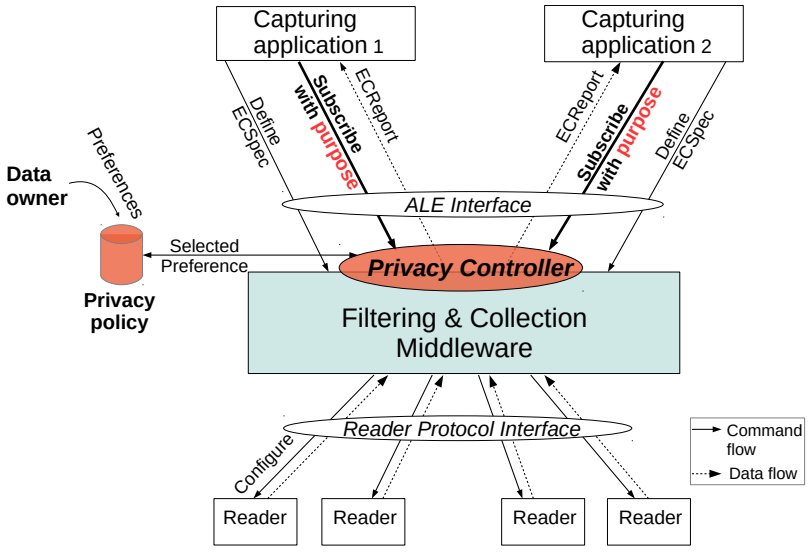


Figure E.3 – Intégration du module de contrôle de la vie privée

Un prototype implémenté dans l'infrastructure Fosstrak

Un prototype a été développé pour illustrer la faisabilité de l'approche et a été appliqué à l'infrastructure Fosstrak, une implémentation open-source des spécifications de l'architecture EPCglobal (se référer à [173, 172]). Quelques tests de performances ont été réalisés sur la nouvelle plateforme. Les résultats de nos tests ont montré que le temps d'exécution dans la platforme incluant notre module de control de la vie privée augmente légèrement par rapport au *middleware* d'origine, lorsque le nombre d'abonnements simultanées est faible. L'écart devient visible seulement à charge élevée (par exemple, lorsque le nombre d'abonnements simultanés et les mises à jour des requêtes ECSpec est élevé).

Conclusion et Perspectives

Bien que la prolifération de la technologie RFID offre plusieurs avantages, son déploiement présente encore des implications sur la vie privée des porteurs des étiquettes RFID. Ces implications sont importantes lorsqu'elles sont liées aux informations personnelles des porteurs (par exemple, les données médicales sur les patients dans le domaine de la santé). Dans ce contexte, le premier objectif de cette thèse est de maintenir une communication sécurisée entre les lecteurs RFID et les étiquettes EPC Gen2. Par une communication sécurisée, nous entendons la sécurité contre les menaces de divulgation de données en présence d'un adversaire fort. Cette communication entre lecteur et étiquette est un réel problème dans les systèmes RFID à faible coût, qui doivent rester pas chers, tout en offrant des protocoles sécurisés correspondants aux ressources limitées de l'étiquette. Une attention particulière est faite sur le système RFID EPC Gen2 dont le modèle de sécurité pour l'accès aux données des étiquettes est vulnérable à des modèles d'attaques basiques. Nous avons proposé de ce fait, un protocole d'établissement et de dérivation de clés dans les systèmes Gen2, qui gère le problème de l'accès à la mémoire de l'étiquette RFID, afin de prévenir les fuites de données. Le protocole, nommé KEDGEN2 a été spécifié en utilisant le langage HLPSL (High Level Protocol Specification Language) et des propriétés de sécurité (authentification forte, *forward secrecy* et *backward secrecy* pour la persistance de la confidentialité) ont été prouvées grâce à des techniques de *model checking* via CL-AtSe (Constraint-Logic based Attack Searcher). Le mécanisme de dérivation de clés pseudo-alétaoires repose sur une adaptation du système de chiffrement Solitaire. La deuxième partie de la thèse s'est focalisée sur la collecte des informations des étiquettes à travers l'intergiciel RFID (appelé, *middleware*). Ce composant se situe entre les lecteurs d'étiquettes et les bases de données applicatives. Il est en charge de collecter, filtrer et agréger des événements issus d'environnements RFID hétérogènes. L'approche que nous avons proposée est centrée sur l'utilisateur, qui peut définir la politque de vie privée s'appliquant aux données collectées et agrégées par le *middleware*. Elle intégre les élémenst suivants: (i) mise en oeuvre d'une politique de vie privée sans interférer avec l'interface standard du *middleware*, (ii) expression des préférences de vie privée en utilisant

le modèle PrivOrBAC et (iii) prise en compte des dimensions d'objectif déclaré de la requête, de précision des résultats produits et de consentement explicite de l'utilisateur. Un prototype a été développé pour illustrer la faisabilité de l'approche et a été appliqué à l'infrastructure Fosstrak, une implémentation open-source des spécifications de l'architecture EPCglobal.

Les perspectives qui peuvent construire des travaux futurs à cette thèse sont:

- Vérifier les propriétés de sécurité du protocole KEDGEN2 avec différents outils et techniques de *model checking*.

- Trouver le seuil optimal pour mettre à jour l'état initial de Solitaire.

- Réaliser un établissement de clé initiale depuis une source alétoire.

- Traiter la propriété d'intraçabilité sur les étiquettes RFID.

- Traiter le contrôle de la vie privée en agrégeant plusieurs événements du *middleware*.

List of Publications

Journal Papers

- W. Tounsi, N. Cuppens-Boulahia, J. Garcia-Alfaro, Y. Chevalier, F. Cuppens, KEDGEN2: A key establishment and derivation protocol for EPC Gen2 RFID systems, *Elsevier Journal of Network and Computer Applications (JNCA)*, March 2014.
- W. Tounsi, N. Cuppens-Boulahia, F. Cuppens, Access and Privacy Control Enforcement in RFID Middleware Systems: Proposal and Implementation on the Fosstrak Platform, *Springer World Wide Web Journal special issue on Large-scale Web Applications in Virtualized Environments*, (Submitted).

International Conference Papers

- W. Tounsi, B. Justus, N. Cuppens-Boulahia, F. Cuppens, J. Garcia-Alfaro, Solitaire Keystream Algorithm as a PRNG for RFID systems, Submitted to *the 8th International Conference on Software Security and Reliability (SERE'14)*, June 2014, San Francisco, California, USA
- W. Tounsi, N. Cuppens-Boulahia, F. Cuppens, J. Garcia-Alfaro, Fine-Grained Privacy Control for the RFID Middleware of EPCglobal Networks, *5th ACM Conference on Management of Emergent Digital EcoSystems (ACM MEDES'13)*, October 2013, Luxembourg **- Selected for possible publication in the springer WWW journal special issue on Large-scale Web Applications in Virtualized Environments.**
- W. Tounsi, N. Cuppens-Boulahia, F. Cuppens, J. Garcia-Alfaro, Privacy-enhanced Filtering and Collection Middleware in EPCglobal Networks, *8th International Conference on Risks and Security of Internet and Systems (CRiSIS'13)*, October 2013, La Rochelle, France
- W. Tounsi, N. Cuppens-Boulahia, F. Cuppens, J. Garcia-Alfaro, Formal verification of a key establishment protocol for EPC Gen2 RFID systems: work in progress. *4th Canada-France MITACS Symposium on Foundations & Practice of Security (FPS'11)*, May 2011, Paris, France
- W. Tounsi, J. Garcia-Alfaro, N. Cuppens-Boulahia, F. Cuppens, Securing the communications of home health care systems based on RFID Sensor networks, *8th International Conference on Communications Networks and Services Research (CNSR'10)*, May 2010, Montréal, Canada

National Conference Papers

- W. Tounsi, J. Garcia-Alfaro, N. Cuppens-Boulahia, F. Cuppens, Protocoles d'échange de clés pour des systèmes de surveillance à base de radio-étiquettes, *5th Conference on Network Architectures and Information Systems (SAR-SSI'10)*, May 2010, Rocquebrune, France, 2010
- W. Tounsi, J. Garcia-Alfaro, N. Cuppens-Boulahia, F. Cuppens, Sécuriser les communications dans les systèmes de surveillance médicale - Protocoles appliqués aux étiquettes à faible coût. *18ème congrès INFORSID'10*, May 2010, Marseille, France, 2010

Bibliography

[1] GSL - GNU Scientific Library. http://www.gnu.org/software/gsl/. 45

[2] Resolution on radio frequency identification. In *25th International Conference of Data Protection and Privacy Commissioners*, 2003. 90

[3] Open entreprise service bus community: documentation, 2011. 76

[4] I. 18000-6. Radio frequency identification for item management - parameters for air interface communications at 860 mhz to 960 mhz. Technical report, International Organization for Standardization (ISO), 2006. 19

[5] M. Abadi. Secrecy by typing in security protocols. *Journal of the ACM*, 46(5):749–786, 1999. 28

[6] M. Abdalla and M. Bellare. Increasing the lifetime of a key: a comparative analysis of the security of re-keying techniques. *Advances in Cryptology (ASIACRYPT'00)*, 1976:546–559, 2000. 35, 123

[7] R. Agrawal, A. Cheung, K. Kailing, and S. Schonauer. Towards traceability across sovereign, distributed RFID databases. In *10th International Database Engineering and Applications Symposium, (IDEAS'06)*, pages 174–184. IEEE, 2006. 80

[8] N. Ajam, N. Cuppens-Boulahia, and F. Cuppens. Contextual privacy management in extended role based access control model. *Data Privacy Management and Autonomous Spontaneous Security*, pages 121–135, 2010. 7, 83, 86, 99

[9] M. E. Ajana, M. Boulmalf, H. Harroud, and H. Hamam. A policy based event management middleware for implementing rfid applications. In *International Conference on Wireless and Mobile Computing, Networking and Communications, (WIMOB'09)*, pages 406–410. IEEE, 2009. 80

[10] A. Ajit, A. L. Denise, and M. E. Johnson. HIPAA Compliance: An Examination of Institutional and Market Forces (1,2). 2009. 4, 141

[11] Architecture Review Committee. The EPCglobal Architecture Framework. Technical report, EPCGlobal, 2010. 2, 74, 75, 126, 136, 164

[12] A. Armando, W. Arsac, T. Avanesov, M. Barletta, A. Calvi, A. Cappai, R. Carbone, Y. Chevalier, L. Compagna, J. Cuéllar, G. Erzse, S. Frau, M. Minea, S. Modersheim, D. Oheimb, G. Pellegrino, S. Elisa Ponta, M. Rocchetto, M. Rusinowitch, M. Torabi Dashti, M. Turuani, and L. Viganò. The AVANTSSAR platform for the automated validation of trust and security of service-oriented architectures. In *18th international conference on Tools and Algorithms for the Construction and Analysis of Systems*, (TACAS'12), pages 267–282. Springer, 2012. 52

[13] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P. Drielsma, P. Heám, O. Kouchnarenko, J. Mantovani, et al. The AVISPA tool for the automated validation of internet security protocols and applications. In *17th International Conference on Computer Aided Verification*, (CAV'05), pages 135–165. Springer, 2005. 28, 52, 53

[14] A. Armando and L. Compagna. SATMC: A SAT-based model checker for security protocols. *Logics in Artificial Intelligence*, pages 730–733, 2004. 28, 52

[15] C. Arora and M. Turuani. Validating Integrity for the Ephemerizer's Protocol with CL-AtSe. In *Formal to Practical Security*, 2009. 52

[16] M. Asadpour and M. Torabi Dashti. A Privacy-friendly RFID Protocol using Reusable Anonymous Tickets. In *10th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, TrustCom'11, pages 206–213. IEEE, 2011. 25, 61, 62, 63

[17] G. Ateniese, J. Camenisch, and B. de Medeiros. Untraceable RFID tags via insubvertible encryption. In *12th ACM conference on Computer and communications security*, pages 92–101. ACM, 2005. 61

[18] B. Barak and S. Halevi. A model and architecture for pseudo-random generation with applications to/dev/random. In *12th ACM conference on Computer and communications security*, (CCS'05), pages 203–212. ACM, 2005. 19

[19] D. Basin, S. Mödersheim, and L. Vigano. OFMC: A symbolic model checker for security protocols. *International Journal of Information Security*, 4:181–208, 2005. 28, 52

[20] L. Batina, J. Guajardo, T. Kerins, N. Mentens, P. Tuyls, and I. Verbauwhede. Public-key cryptography for RFID-tags. In *Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops, 2007. PerCom Workshops' 07*, pages 217–222. IEEE, 2007. 17

[21] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation. In *38th Annual Symposium on Foundations of Computer Science*, (FOCS'97), pages 394–403, 1997. 35, 49, 69, 122

[22] M. Bellare, T. Krovetz, and P. Rogaway. Luby-Rackoff backwards: Increasing security by making block ciphers non-invertible. *Advances in Cryptology (EUROCRYPT'98)*, 1403:266–280, 1998. 122, 123

[23] M. Bellare and B. Yee. Forward-security in private-key cryptography. *Topics in Cryptology–(CT-RSA'03)*, 2612:1–18, 2003. 19

[24] J. Biskup and J.-H. Lochner. Enforcing confidentiality in relational databases by reducing inference control to access control. In *Information Security*, pages 407–422. Springer, 2007. 82

[25] B. Blanchet. Security protocol verification: Symbolic and computational models. In *Proceedings of the First international conference on Principles of Security and Trust*, pages 3–29. Springer-Verlag, 2012. 26, 29

[26] A. Bogdanov, G. Leander, C. Paar, A. Poschmann, M. Robshaw, and Y. Seurin. Hash Functions and RFID Tags : Mind The Gap. In *10th International Workshop Cryptographic Hardware and Embedded Systems*, (CHES'08), 2008. 4, 18, 24, 62, 139

[27] L. Bolotnyy and G. Robins. Physically unclonable function-based security and privacy in RFID systems. In *International Conference on Pervasive Computing and Communications (PerCom 2007)*, pages 211–220. IEEE Press, 2007. 24

[28] C. Boult, M. Altmann, D. Gilbertson, C. Yu, and R. Kane. Decreasing disability in the 21st century: the future effects of controlling six fatal and nonfatal conditions. *American Journal of Public Health*, 86(10):1388–1393, 1996. 1, 136

[29] J. Bringer and H. Chabanne. Trusted-HB: A Low-Cost Version of HB+ Secure Against Man-in-the-Middle Attacks. *IEEE Transactions on Information Theory*, 54:4339–4342, 2008. 25

[30] J. Bringer, H. Chabanne, and E. Dottax. HB++: a Lightweight Authentication Protocol Secure against Some Attacks. In *Second International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing*, (SecPerU'06), pages 28–33. IEEE, 2006. 25

[31] M. Brusò, K. Chatzikokolakis, and J. den Hartog. Formal Verification of Privacy for RFID Systems. In *21th Computer Security Foundations Symposium (CSF'10)*, pages 75–88. IEEE, 2010. 25, 61, 63

[32] M. Burmester and J. Munilla. Lightweight RFID authentication with forward and backward security. *ACM Transactions on Information and System Security (TISSEC'11*, 14(1):11, 2011. 25, 61, 62, 63

[33] T. Cao, E. Bertino, and H. Lei. Security analysis of the SASI protocol. *IEEE Transactions on Dependable and Secure Computing*, 6:73–77, 2009. 25

[34] M. Chaudhry, Q. Ahmad, I. Sarwar, and A. H. Akbar. Comparative study of RFID middlewares-defining the roadmap to SOA-based middlewares. In *International Conference on Industrial Technology (ICIT'10)*, pages 1386–1393. IEEE, 2010. 5, 88, 142

[35] W. Che, H. Deng, W. Tan, and J. Wang. A random number generator for application in RFID tags. In *Networked RFID systems and lightweight cryptography*, pages 279–287. Springer, 2008. 21

[36] Y. Chevalier, L. Compagna, J. Cuellar, P. Drielsma, J. Mantovani, S. Moedersheim, and L. Vigneron. A high level protocol specification language for industrial security-sensitive protocols. In *Proceedings of Workshop on Specification and Automated Processing of Security Requirements*, volume 180 of *(SAPS'04)*, 2004. 53

[37] Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP decision procedure for protocol insecurity with XOR. *Theoretical Computer Science*, 338(1):247–274, 2005. 28, 52

[38] H.-Y. Chien. SASI: A New Ultralightweight RFID Authentication Protocol Providing Strong Authentication and Strong Integrity. *IEEE Transactions on Dependable and Secure Computing*, 4:337–340, 2007. 25

[39] F. Christian. Integrating RFID Readers in the Enterprise IT–Overview of Intra-organizational RFID System Services and Architectures. *Academic publication of the Auto-ID Labs*, 2008. 5, 76, 88, 142

[40] F. Christian, R. Christof, and L. Matthias. RFID Application Development with the Accada Middleware Platform. *IEEE Systems Journal*, 1(2):82–94, 2007. 103

[41] F. Christian, S. Roland, and L. Marc. Scanning with a purpose–supporting the fair information principles in rfid protocols. In *Ubiquitous Computing Systems*, pages 214–231. Springer, 2005. 80, 88, 89

[42] C. Cid, S. Murphy, and M. J. B. Robshaw. *Algebraic aspects of the advanced encryption standard*. Springer, 2006. 43

[43] P. H. Cole and D. C. Ranasinghe. *Networked RFID systems and lightweight cryptography: raising barriers to product counterfeiting*. Springer, 2008. 12, 14

[44] Commission of the European Communities. Commission Recommendation on the implementation of privacy and data protection principles in applications supported by radio-frequency identification. 2009. 4, 82, 83, 90, 141

[45] H. Comon-Lundh and V. Cortier. Security properties: two agents are sufficient. *Science of Computer Programming*, 50(1):51–71, 2004. 28

[46] H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *18th Annual IEEE Symposium on Logic in Computer Science,*, pages 271–280. IEEE, 2003. 28

[47] V. Cortier, S. Kremer, and B. Warinschi. A survey of symbolic methods in computational analysis of cryptographic systems. *Journal of Automated Reasoning*, 46(3-4):225–259, 2011. 26

[48] C. J. F. Cremers, P. Lafourcade, and P. Nadeau. Comparing State Spaces in Automatic Security Protocol Analysis. In *Formal to Practical Security*, pages 70–94, 2009. 52

[49] F. Cuppens and N. Cuppens-Boulahia. Modeling contextual security policies. *International Journal of Information Security*, 7(4):285–305, 2008. 84, 85

[50] E. Damiani, S. Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati. Balancing confidentiality and efficiency in untrusted relational DBMSs. In *10th ACM conference on Computer and communications security*, pages 93–102. ACM, 2003. 81

[51] H. E. Daniel, B. P. Wayne, and F. Kevin. Initial SRAM state as a fingerprint and source of true random numbers for RFID tags. In *Conference on RFID Security*, volume 7, 2007. 21

[52] D. Danny and Y. Andrew. On the security of public key protocols. *Transactions on Information Theory*, 29(2):198–208, 1983. 27

[53] G. Denker, J. Meseguer, and C. Talcott. Protocol specification and analysis in maude. In *Workshop on Formal Methods and Security Protocols*, volume 25. Citeseer, 1998. 28, 52

[54] A. Desai, A. Hevia, and Y. Yin. A practice-oriented treatment of pseudorandom number generators. In *Advances in Cryptology*, (EUROCRYPT'02), pages 368–383. Springer, 2002. 19

[55] T. Dimitriou. rfidDOT: RFID delegation and ownership transfer made simple. In *Proceedings of the 4th international conference on Security and privacy in communication netowrks*, page 34. ACM, 2008. 61

[56] D. Duc and K. Kim. Securing HB+ against GRS man-in-the-middle attack. In *Symposium on Cryptography and Information Security*, (SCIS'07), 2007. 25

[57] G. Eberhard and M. Markus. Fine-grained access control for epc information services. In *The Internet of Things*, pages 35–49. Springer, 2008. 80

[58] ECRYPT Network of Excellence. The estream project. 21

[59] J. Edney and W. Arbaugh. *Real 802.11 security: Wi-Fi protected access and 802.11i*. Addison-Wesley Professional, 2004. 4, 140

[60] B. Eggen, G. Hollemans, and R. van de Sluis. Exploring and enhancing the home experience. *Cognition, Technology & Work*, 5:44–54, 2003. 1, 136

[61] EPCglobal. EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860-960 MHz. Technical report, Version 1.2.0, http://www.epcglobalinc.org/standards/, 2008. 2, 11, 12, 14, 15, 136

[62] EPCglobal. Inc. The Application-Level Events (ALE) Specification, version 1.1 - Part 1: Core Specification. Technical report, EPCglobal, 2008. 76, 78, 80, 93

[63] EPCglobal Inc. Low Level Reader Protocol (LLRP). Technical Report Version 1.1, EPCglobal, 2010. 104, 109

[64] EPCglobal. Inc. Public Policy. Technical report, 2011. 90

[65] EPCglobal Inc. The EPCglobal Website. http://www.gs1.org/epcglobal, 2013. 2, 6, 11, 119, 136

[66] S. Escobar, C. Meadows, and J. Meseguer. Maude-NPA: Cryptographic Protocol Analysis Modulo Equational Properties. In *FOSAD*, pages 1–50, 2007. 52

[67] O. for Economic Co-operation & Development Council. *Recommendation of the Council Concerning Guidelines Governing the Protection of Privacy and Transborder Flows of Personal Data*. OECD, 1980. 5, 81, 82, 83, 88, 90, 141

[68] S. Foresti. *Preserving Privacy in Data Outsourcing*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010. 81

[69] Fosstrak. Project License. Technical report, 2009. 6, 80, 103, 104, 119, 164

[70] F. D. Garcia, G. de Koning Gans, R. Muijrers, P. Van Rossum, R. Verdult, R. W. Schreur, and B. Jacobs. Dismantling MIFARE classic. In *Computer Security*, (ESORICS'08), pages 97–114. Springer, 2008. 4, 16, 20, 139

[71] J. Garcia-Alfaro, J. Herrera-Joancomartí, and J. Melià-Seguí. Practical Eavesdropping of Control Data from EPC Gen2 Queries with a Programmable RFID Toolkit. *Hakin9*, 2011. [Online]. 32

[72] H. Gilbert, M. Robshaw, and Y. Seurin. HB#: Increasing the Security and Efficiency of HB+. In *27th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, (EUROCRYPT'08), pages 361–378. Springer-Verlag, 2008. 25

[73] H. Gilbert, M. Robshaw, and H. Sibert. Active attack against HB+: a provably secure lightweight authentication protocol. *Electronics Letters*, 41:1169–1170, 2005. 25

[74] A. Gildas and O. Oechslin, Philippe. A Scalable and Provably Secure Hash Based RFID Protocol. In *International Workshop on Pervasive Computing and Communication Security*, (PerSec'05). IEEE, 2005. 23

[75] GS1 AISBL. GS1 EPCglobal Tag Data Translation (TDT) 1.6. Technical report, EPCGlobal, 2011. 12, 75, 125

[76] GS1 AISBL. GS1 EPC Tag Data Standard 1.7. Technical report, EPCglobal, 2013. 12, 92

[77] G. Hammouri and B. Sunar. PUF-HB: A Tamper-Resilient HB Based Authentication Protocol. In *Applied Cryptography and Network Security (ACNS'08)*, 2008. 25

[78] Y. Hanatani, M. Ohkubo, S. Matsuo, K. Sakiyama, and K. Ohta. A Study on Computational Formal Verification for Practical Cryptographic Protocol: The Case of Synchronous RFID Authentication. In *Financial Cryptography Workshops*, pages 70–87, 2011. 25, 61, 63

[79] D. Henrici and P. Müller. Hash-based enhancement of location privacy for radio-frequency identification devices using varying identifiers. In *Second Annual Conference on Pervasive Computing and Communications Workshops*. IEEE, 2004. 23

[80] D. Henrici and P. Müller. Providing Security and Privacy in RFID Systems Using Triggered Hash Chains. In *Sixth Annual IEEE International Conference on Pervasive Computing and Communications*, PerCom'08, 2008. 23

[81] N. Hopper and M. Blum. Secure human identification protocols. In *7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT'01, pages 52–66. Springer, 2001. 24

[82] K.-T. Huang, P.-Y. Lin, C.-Y. Chiang, J.-S. Chang, C.-N. Huang, and C.-T. Chan. An Intelligent RFID System for Improving Elderly Daily Life Independent in Indoor Environment. In *Smart Homes and Health Telematics*, 2008. 1, 117, 135

[83] IBM Corp. IBM WebSphere Premises Server. http://www-01.ibm.com/software/integration/sensor-events/, 2010. 80

[84] INRIA. ASPIRE-Advanced Sensors and lightweight Programmable middleware for Innovative RFID Enterprise applications. Technical report, 2009. 80

[85] A. Ismael, C. Carlos, C. Jose, H. Rubén, and V. Enrique. Managing RFID Sensors Networks with a General Purpose RFID Middleware. *Sensors*, 12(6):7719–7737, 2012. 80

[86] W. Jakkhupan, S. SArch-int, and Y. Li. Business process analysis and simulation for the RFID and EPCglobal Network enabled supply chain: A proof-of-concept approach. *Journal of Network and Computer Applications*, 34(3):949–957, 2011. 11

[87] A. Joux. *Algorithmic cryptanalysis*. CRC Press, 2009. 20

[88] A. Juels. RFID security and privacy: a research survey. *Journal of Selected Areas in Communications*, 24:381–394, 2006. 17, 80

[89] A. Juels, R. Pappu, and B. Parno. Unidirectional key distribution across time and space with applications to rfid security. In *USENIX Security Symposium*, pages 75–90, 2008. 31

[90] A. Juels, R. L. Rivest, and M. Szydlo. The blocker tag: selective blocking of RFID tags for consumer privacy. In *10th ACM conference on Computer and communications security*, pages 103–111. ACM, 2003. 17

[91] A. Juels and S. Weis. Authenticating pervasive devices with human protocols. In *25th Annual International Cryptology Conference*, CRYPTO'05, 2005. 5, 24, 40, 49, 139

[92] A. A. E. Kalam, S. Benferhat, A. Miège, R. E. Baida, F. Cuppens, C. Saurel, P. Balbiani, Y. Deswarte, and G. Trouessin. Organization based access control. In *POLICY*. 4th IEEE International Workshop on Policies for Distributed Systems and Networks, 2003. 83

[93] S. Kardas, S. Celik, M. Yildiz, and A. Levi. PUF-enhanced offline RFID security and privacy. *Journal of Network and Computer Applications*, 35(6):2059–2067, 2012. 24

[94] S. Kartakis, V. Sakkalis, P. Tourlakis, G. Zacharioudakis, and C. Stephanidis. Enhancing health care delivery through ambient intelligence applications. *Sensors*, 12:11435–11450, 2012. 1, 117, 135

[95] H. Kim, J. Oh, J. Kim, Y. Jeong, and J. Choi. Formal Verification of Cryptographic Protocol for Secure RFID System. In *4th International Conference on Networked Computing and Advanced Information Management*, pages 470–477. IEEE, 2008. 25, 61, 63

[96] P. Kitsos and Y. Zhang. *RFID security: techniques, protocols and system-on-chip design*. Springer Verlag, 2008. 22

[97] F. Klaus. Known attacks on RFID systems, possible countermeasures and upcoming standardisation activities. In *5th European Workshop on RFID Systems and Technologies*, 2009. 39

[98] M. O. Koutarou, K. Suzuki, and S. Kinoshita. Cryptographic Approach to "Privacy-Friendly" Tags. In *RFID Privacy Workshop*, 2003. 61

[99] S. Kremer. *Modelling and analyzing security protocols in cryptographic process calculi*. PhD thesis, École normale supérieure de Cachan-ENS Cachan, 2011. 6, 52, 141

[100] R. Küsters and T. Truderung. Using ProVerif to Analyze Protocols with Diffie-Hellman Exponentiation. *22nd Computer Security Foundations Symposium (CSF'09)*, 9:157–171, 2009. 28

[101] R. Küsters and T. Truderung. Reducing Protocol Analysis with XOR to the XOR-Free Case in the Horn Theory Based Approach. *Journal of Autom. Reasoning*, 46(3-4):325–352, 2011. 52

[102] L. Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems (TOPLAS'94)*, 16:872–923, 1994. 53, 54

[103] X. Leng, K. Mayes, and K. Markantonakis. HB-MP+ protocol: An improvement on the HB-MP protocol. In *International Conference on RFID*, pages 118–124. IEEE, 2008. 25

[104] T. Li and G. Wang. Security Analysis of Two Ultra-Lightweight RFID Authentication Protocols. In *22nd International Information Security Conference*, (IFIP SEC'07), 2007. 25

[105] C. H. Lim and T. Kwon. Strong and robust RFID authentication enabling perfect ownership transfer. In *Conference on Information and Communications Security*, (ICICS'06), 2006. 60, 61

[106] S. Lim and I. Yie. Probabilistic privacy leakage from challenge-response RFID authentication protocols. In *Proceedings of the 7th Conference on 7th WSEAS International Conference on Applied Informatics and Communications (AIC'07)*, volume 7, pages 285–288, Stevens Point, Wisconsin, USA, 2007. 24

[107] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 147–166. Springer, 1996. 28, 52

[108] G. Lowe. Towards a completeness result for model checking of security protocols. *Journal of computer security*, 7(2):89–146, 1999. 28

[109] D. Mahoney. The aging nurse workforce and technology. *Gerontechnology*, 10:13–25, 2011. 1, 117, 136

[110] P. Manzanares-Lopez, J. P. Muñoz-Gea, J. Malgosa-Sanahuja, and J. C. Sanchez-Aarnoutse. An efficient distributed discovery service for EPCglobal network in nested package scenarios. *Journal of Network and Computer Applications*, 34(3):925–937, 2011. 12

[111] H. Marit, S. Ari, and C. Alissa. Privacy and identity management. *IEEE Security and Privacy*, 6(2):38–45, Mar. 2008. 81

[112] F. Martin and C. Rechberger. A Case Against Currently Used Hash Functions in RFID Protocols. In *On the Move to Meaningful Internet Systems*, (OTM'06), 2006. 18, 24, 62

[113] H. Martin, J. Thomas, and M. Willi. Grain: a stream cipher for constrained environments. *Journal of Wireless and Mobile Computing*, 2(1):86–93, 2007. 21

[114] A. Masoumzadeh and J. Joshi. PuRBAC: Purpose-aware role-based access control. *On the Move to Meaningful Internet Systems (OTM'08)*, pages 1104–1121, 2008. 83

[115] J. Massey. Shift-register synthesis and BCH decoding. *Information Theory, IEEE Transactions on*, 15(1):122–127, 1969. 20

[116] C. Meadows. Open issues in formal methods for cryptographic protocol analysis. In *DARPA Information Survivability Conference and Exposition, 2000. (DISCEX'00)*, volume 1, pages 237–250. IEEE, 2000. 27

[117] C. Meadows. Formal methods for cryptographic protocol analysis: Emerging issues and trends. *Journal on Selected Areas in Communications*, 21(1):44–54, 2003. 26

[118] W. Meier and O. Staffelbach. The self-shrinking generator. In *Communications and Cryptography*, pages 287–295. Springer, 1994. 20

[119] J. Melia-Segui, J. Garcia-Alfaro, and J. Herrera-Joancomarti. Analysis and improvement of a pseudorandom number generator for epc gen2 tags. In *Financial Cryptography and Data Security*, pages 34–46. Springer, 2010. 21

[120] J. Melia-Segui, J. Garcia-Alfaro, and J. Herrera-Joancomarti. A practical implementation attack on weak pseudorandom number generator designs for EPC Gen2 tags. *Wireless Personal Communications*, 59(1):27–42, 2011. 3, 15, 138

[121] A. Menezes, P. Van Oorschot, and S. Vanstone. *Handbook of applied cryptography*. CRC Press Inc., 1997. 16, 17, 19, 22, 23, 34

[122] M. J. Mihaljević. A faster cryptanalysis of the self-shrinking generator. In *Information security and privacy*, pages 182–189. Springer, 1996. 20

[123] B. Mihály, B. Balázs, L. Péter, L. Krisztina, and N. Dániel. Breaking EMAP. In *Conference on Security and Privacy for Communication Networks (SecureComm'07)*, 2007. 25

[124] B. Mihály, B. Balázs, L. Péter, L. Krisztina, and N. Dániel. Breaking LMAP. In *Conference on RFID Security*, 2007. 25

[125] B. Mihály, B. Balázs, L. Péter, L. Krisztina, and N. Dániel. Passive Attack Against the M2AP Mutual Authentication Protocol for RFID Tags. In *First International EURASIP Workshop on RFID Technology*, 2007. 25

[126] J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *8th ACM conference on Computer and Communications Security*, pages 166–175. ACM, 2001. 28

[127] G. A. Miller. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological review*, 63(2):81, 1956. 40

[128] J. C. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using murφ. In *Symposium on Security and Privacy*, pages 141–151. IEEE, 1997. 28, 52

[129] A. Mitrokotsa, C. Onete, and S. Vaudenay. Mafia Fraud Attack against the RČ Distance-Bounding Protocol. In *International Conference on RFID-Technology and Applications (RFID-TA'12)*, Nice, France, November 2012. IEEE. 24

[130] Motorola. RFID technology and EPC in retail. Technical report, Symbol Technologies, 2004. 1, 135

[131] J. Munilla and A. Peinado. HB-MP: A further step in the HB-family of lightweight authentication protocols. *Computer Networks*, 51:2262–2267, 2007. 25

[132] D. Nancy, L. Patrick, M. John, and S. Andre. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security*, 12(2):247–311, 2004. 28

[133] R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978. 27

[134] Q. Ni, D. Lin, E. Bertino, and J. Lobo. Privacy-Aware Role Based Access Control. In *12th symposium on Access control models and technologies*, pages 41–50. ACM, 2007. 83

[135] Official Journal of the European Communities, editor. *Directive 95/46/EC of the European Parliament and of the Council on the protection of Individuals with regard to the processing of personal data and on the free movement of such data*, number 281 in 31, 1995. 80, 82, 90

[136] M. Ohkubo, K. Suzuki, and S. Kinoshita. Efficient Hash-Chain Based RFID Privacy Protection Scheme. In *International Conference on Ubiquitous Computing (UbiComp'04), Workshop Privacy: Current Status and Future Directions*, 2004. 23

[137] P. on a Research Agenda and N. D. for an Aging World. *Preparing for an aging world: the case for cross-national research*. National Academy Press, 2001. 1, 136

[138] Oracle. Oracle Application Server Wireless. Technical Report 10.1.2, 2005. 80, 91, 164

[139] S. Oulmakhzoune, N. Cuppens-Boulahia, F. Cuppens, S. Morucci, M. Barhamgi, and D. Benslimane. Privacy query rewriting algorithm instrumented by a privacy-aware access control model. In *Annals of telecommunications (ANTE'13)*, 2013. 110

[140] G. P. Hancke. Design of a secure distance-bounding channel for RFID. *Journal of Network and Computer Applications*, 34(3):877–887, 2011. 24

[141] C. Paar, A. Poschmann, and M. Robshaw. New designs in lightweight symmetric encryption. In *RFID Security*, pages 349–371. Springer, 2009. 18

[142] G. Pallapa and S. K. Das. Challenges of designing privacy enhanced context-aware middleware for assisted healthcare. In *Smart Homes and Health Telematics*, pages 200–207. Springer, 2008. 81, 82

[143] R. Pappu. *Physical One-Way Functions*. PhD thesis, Massachusetts Institute of Technology (MIT), 2001. 24

[144] L. C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of computer security*, 6(1):85–128, 1998. 27

[145] P. Peris-Lopez, J. Hernandez-Castro, J. Estevez-Tapiador, and A. Ribagorda. LAMED a PRNG for EPC Class-1 Generation-2 RFID specification. *Computer Standards & Interfaces*, pages 88–97, 2007. 15, 21

[146] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, and A. Ribagorda. EMAP: An Efficient Mutual Authentication Protocol for Low-Cost RFID Tags. In *On the Move to Meaningful Internet Systems: Federated Conferences and Workshop*, (OTM'06), 2006. 25

[147] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, and A. Ribagorda. LMAP: A Real Lightweight Mutual Authentication Protocol for Low-cost RFID tags. In *Workshop on RFID Security*, (RFIDSec'06), 2006. 25

[148] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, and A. Ribagorda. M2AP: A Minimalist Mutual-Authentication Protocol for Low-cost RFID Tags. In *International Conference on Ubiquitous Intelligence and Computing*, (UIC'06), 2006. 25

[149] R. D. Pietro and R. Molva. An optimal probabilistic solution for information confinement, privacy, and security in RFID systems. *Journal of Network and Computer Applications*, 34(3):853–863, 2011. 24

[150] S. Piramuthu and Y. Tu. Modified HB authentication protocol. In *Western European Workshop on Research in Cryptology*, (WEWoRC'07), 2007. 25

[151] B. Pogorelov and M. Pudovkina. Properties of the transformation semigroup of the solitaire stream cipher. *IACR Cryptology ePrint Archive*, 2003:169, 2003. 43

[152] B. Prabhu, X. Su, H. Ramamurthy, C.-C. Chu, and R. Gadh. WinRFID: A Middleware for the Enablement of Radio frequency Identification (RFID)-Based Applications. *Mobile, wireless, and sensor networks: Technology, applications, and future directions*, page 313, 2006. 80

[153] F. Rizzo, M. Barboni, L. Faggion, G. Azzalin, and M. Sironi. Improved security for commercial container transports using an innovative active RFID system. *Journal of Network and Computer Applications*, 34(3):846–852, 2011. 11

[154] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, and E. Barker. A statistical test suite for random and pseudorandom number generators for cryptographic applications. Technical report, DTIC Document, 2001. 64

[155] M. Rusinowitch and M. Turuani. Protocol insecurity with a finite number of sessions and composed keys is NP-complete. *Theoretical Computer Science*, 299(1):451–475, 2003. 28

[156] A.-R. Sadeghi, I. Visconti, and C. Wachsmann. Anonymizer-enabled security and privacy for RFID. In *Cryptology and Network Security*, pages 134–153. Springer, 2009. 61

[157] M. Safkhani, N. Bagheri, M. Naderi, and S. Sanadhya. Security analysis of LMAP++, an RFID authentication protocol. *International Conference for Internet Technology and Secured Transactions (ICITST'11)*, 2011:689–694, 2011. 25

[158] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996. 83

[159] M. Schapranow, A. Zeier, and H. Plattner. Security Extensions for Improving Data Security of Event Repositories in EPCglobal Networks. In *9th International Conference on Embedded and Ubiquitous Computing (IFIP EUC'11)*, pages 213–220. IEEE, 2011. 80

[160] B. Schneier. Applied cryptography. Protocols, Algorithms, and Source Code in C, 1996. 4, 20, 139

[161] B. Schneier. The Solitaire Encryption Algorithm, version 1.2. https://www.schneier.com/solitaire.html, 1999. 5, 40, 45, 139

[162] S.E. Sarma. Toward the 5 cents tag. Technical report, Auto-ID, 2001. 12

[163] J. M. Seguí. *Lightweight PRNG for Low-cost Passive RFID Security Improvement.* PhD thesis, Universitat Oberta de Catalunya, 2011. 20, 21

[164] C. E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001. 18

[165] B. Song and C. J. Mitchell. RFID authentication protocol for low-cost tags. In *first ACM conference on Wireless network security*, (WiSec'08), 2008. 60

[166] J. Song and H. Kim. The RFID middleware system supporting context-aware access control service. In *The 8th International Conference on Advanced Communication Technology, 2006. (ICACT'06)*, volume 1. IEEE, 2006. 80

[167] J. Song, T. Kim, S. Lee, and H. Kim. Security enhanced RFID middleware system. *World Academy of Science, Engineering and Technology*, 10, 2005. 80

[168] J. Soto. Statistical testing of random number generators. In *22nd National Information Systems Security Conference*, volume 10, page 12. NIST Gaithersburg, MD, 1999. 64

[169] G. Stefanos, S. Diomidis, and G. Panagiotis. Security protocols over open networks and distributed systems: Formal methods for their analysis, design, and verification. *Computer Communications*, 22(8):697–709, 1999. 26

[170] H. . Tore. On functions of linear shift register sequences. In *Advances in Cryptology (EUROCRYPT'85)*, pages 119–129. Springer, 1986. 20

[171] W. Tounsi, N. Cuppens-Boulahia, F. Cuppens, and J. Garcia-Alfaro. Formal Verification of a Key Establishment Protocol for EPC Gen2 RFID Systems: Work in Progress. In *Foundations and Practice of Security*, volume 6888, pages 242–251. 2012. 5, 139

[172] W. Tounsi, N. Cuppens-Boulahia, F. Cuppens, and J. Garcia Alfaro. Fine-Grained Privacy Control for the RFID Middleware of EPCglobal Networks. In *5th ACM Conference on Management of Emergent Digital EcoSystems (MEDES'13)*, pages 60–67, 2013. 6, 143

[173] W. Tounsi, N. Cuppens-Boulahia, F. Cuppens, and J. Garcia-Alfaro. Privacy-enhanced Filtering and Collection Middleware in EPCglobal Networks. In *8th International Conference on Risks and Security of Internet and Systems (CRISIS'13)*, 2013. 6, 143

[174] W. Tounsi, N. Cuppens-Boulahia, J. Garcia-Alfaro, Y. Chevalier, and F. Cuppens. KEDGEN2: A key establishment and derivation protocol for EPC Gen2 RFID systems. *Journal of Network and Computer Applications*, 39(0):152 – 166, 2014. 6, 141

[175] W. Tounsi, J. Garcia-Alfaro, N. Cuppens-Boulahia, and F. Cuppens. Securing the communications of home health care systems based on RFID sensor networks. In *8th Conference on Communications Networks and Services Research (CNSR'10)*, pages 284–291. IEEE, 2010. 5, 80, 139

[176] W. Tounsi, B. Justus, , N. Cuppens-Boulahia, and J. Cuppens, F. Garcia-Alfaro. Solitaire Keystream Algorithm as a PRNG for RFID systems. In *29th International Conference on Systems Security and Privacy Protection (IFIP'SEC'14) (to be submitted)*, 2014. 5, 140

[177] G. Tsudik. YA-TRAP: Yet another trivial RFID authentication protocol. In *4th Conference on Pervasive Computing and Communications Workshops (PerCom'06 Workshops)*, pages 640–643. IEEE, 2006. 24

[178] M. Turuani. The CL-Atse protocol analyser. *Term Rewriting and Applications*, 4098:277–286, 2006. 28, 52

[179] T. Van Le, M. Burmester, and B. De Medeiros. Universally composable and forward-secure RFID authentication and authenticated key exchange. In *2nd symposium on Information, computer and communications security*, pages 242–252. ACM, 2007. 25, 61, 62, 63

[180] P. C. Van Oorschot and M. J. Wiener. On diffie-hellman key agreement with short exponents. In *Advances in Cryptology (EUROCRYPT'96)*, pages 332–343. Springer, 1996. 22

[181] Q. Wang, T. Yu, N. Li, J. Lobo, E. Bertino, K. Irwin, and J.-W. Byun. On the correctness criteria of fine-grained access control in relational databases. In *33rd international conference on Very large data bases*, pages 555–566, 2007. 120, 127, 128

[182] S. Weis, S. Sarma, R. Rivest, and D. Engels. Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. In *International Conference on Security in Pervasive Computing*, (SPC'03), 2003. 23

[183] A. F. Westin. Privacy and freedom. *Washington and Lee Law Review*, 25(1):166, 1968. 81

[184] O. K. Y. Boichut, P.-C. Heam. Automatic Verification of Security Protocols Using Approximations. Technical report, INRIA Research Report, 2005. 52

[185] N. Yang, H. Barringer, and N. Zhang. A purpose-based access control model. In *Third International Symposium on Information Assurance and Security (IAS'07)*, pages 143–148. IEEE, 2007. 83

[186] W. Yao, C.-H. Chu, and Z. Li. Leveraging complex event processing for smart hospitals using RFID. *Journal of Network and Computer Applications*, 34(3):799–810, 2011. 11

List of Figures